เอกสารอ้างอิง

1. Block, F., W.W. Hansen, M.E. Packard, "Nuclear Induction", Phys. Rev., 69, 127, 1946.

2. Purcell, E.M., H.C. Torrey, and R.V. Pound, "Resonance Absorption by Nuclear Magnetic Moment in a Solid", Phys. Rev., 69, 37, 1946.

3. Cho, Z.H., H.S. Kim, H.B. Song, James Cumming, "Fourier Transform Nuclear Magnetic Resonance Tomographic Imaging", Proceeding of the IEEE, 70(10), 1152-1173, 1982.

4. Bottomley, PA, "NMR Imaging Techniques and Application", General Electric Company Corporate Research and Development Schenectady, New York, 1981.

5. Kumar, Anil, Dieter Welti, and Richard R. Ernst, "NMR Fourier Zeugmatography", Journal of Magnetic Resonance, 18, 69-83, 1975.

6. Shaw, Derek, Fourier Transform N.M.R. Spectroscopy, Elsevier Science Publishing B.V., Amsterdam, 1984.

7. Poole, Charles P., Jr, Horacio A. Farach, Relaxation in Magnetic Resonance, Academic Press, New York, 1971.

8. Andrew, E. R., Nuclear Magnetic Resonance, pp. 88-89, The Syndics of the Cambridge University Press, London, 1958.

9. Slichter, Charles P., Principles of Magnetic Resonance, Harper & Row Publishers, New York, 1963.

10. Ralston, Athony, Philip Rabinowitz, A First Course in Numerical Analysis, pp. 263-270, McGraw-Hill Inc., New York, n.d.

11. Princeton Applied Research Co., Model 4202 Signal Averager Operating and Service Manual, U.S.A., 1978.

12. Bracewell, Ronald N., The Fourier Transform and its Applications, McGraw-Hill Inc., New York, n.d.

13. Borland International, Inc, Turbo Pascal Ver. 4.0 Owner's Handbook, U.S.A., 1987

ภาคผนวก ก

วิธีใช้งานและการทำงานของโปรแกรมสำหรับการวิเคราะห์ข้อมูล

ในโครงการวิจัยเรื่องการสร้างภาพด้วยวิธีเอ็นเอ็มอาร์นี้ ส่วนที่เป็นงานของวิทยานิพนธ์นี้ ได้แก่ส่วนของการวิเคราะห์ข้อมูลและการนำผลการวิเคราะห์มาแสดงเป็นภาพ ซึ่งกระบวนการทั้งหมดนี้ทำโดยใช้คอมพิวเตอร์ทั้งสิ้น ซึ่งจำเป็นที่จะต้องเขียนและพัฒนาส่วนโปรแกรมขึ้นมาเองทั้งหมด เพื่อใช้ในการวิจัยเรื่องนี้โดยเฉพาะ ซึ่งการวิเคราะห์ผลนี้จะต้องใช้วิธีการแปลงแบบฟูเรียร์ในสองมิติกับข้อมูลที่ได้มา และเนื่องจากว่าข้อมูลที่ได้มานี้เป็นตัวเลขทั้งสิ้น ดังนั้นจึงได้เลือกนำเทคนิควิธีการแปลงแบบฟูเรียร์อย่างเร็วหรือเอฟเอฟทีมาใช้เพื่อให้การประมวลผลข้อมูลทั้งหมดเป็นไปอย่างรวดเร็ว เนื่องจากข้อมูลมีเป็นจำนวนมาก หากไม่ใช้วิธีนี้แล้วการประมวลผลจะต้องใช้เวลามากดังได้กล่าวถึงไว้ในบทที่ 4

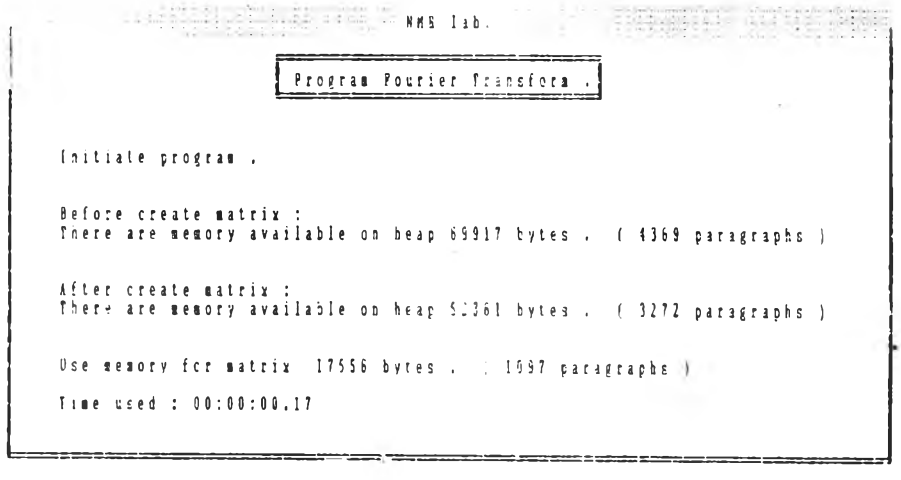
<u>การทำงานของโปรแกรม</u>

โปรแกรมที่พัฒนาขึ้นมานี้เป็นโปรแกรมที่พัฒนาให้ใช้งานได้ง่ายและมีประสิทธิภาพสูง กล่าวคือในการใช้งานผู้ใช้สามารถสั่งงานได้โดยใช้การเลือกจากรายการหรือเมนู (Menu) ซึ่งจะปรากฏขึ้นมาให้เลือกสั่งงานได้ตามความต้องการ และการทำงานของโปรแกรมได้ถูกเลือกใช้วิธีการที่มีความเร็วสูง เช่นการใช้เอฟเอฟที (FFT) การแสดงผลที่เข้าใจได้ง่ายและสามารถใช้งานบนคอมพิวเตอร์ชนิดที่เข้ากันได้กับไอบีเอ็ม พีซี (IBM PC Campatable)ได้ทุกแบบ

โปรแกรมนี้เขียนขึ้นโดยใช้ภาษาปาสคาล (Pascal) และใช้โปรแกรมตัวแปลภาษา (Compiler) ชื่อ เทอร์โบปาสคาล รุ่น 5.0 (Turbo Pascal version 5.0) ผลิตขึ้นโดยบริษัทบอร์แลนด์ (Borland) ซึ่งตัวโปรแกรมที่เขียนขึ้นนี้มีความยาวทั้งหมดประมาณ 7,000 บรรทัด และได้แสดงไว้ในภาคผนวกแล้ว

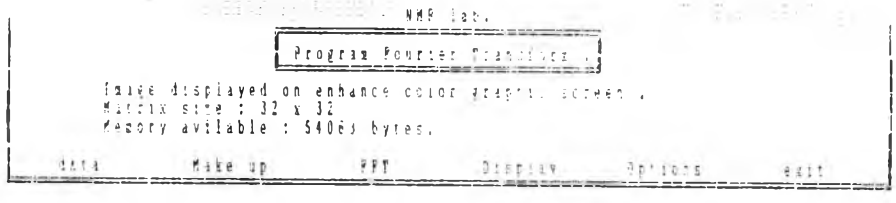
การใช้งานโปรแกรมจะเริ่มที่เรียกโปรแกรมขึ้นมาก่อน เมื่อโปรแกรมเริ่มทำงานก็จะมีการเตรียมเนื้อที่ในหน่วยความจำส่วนหนึ่งไว้ใช้เก็บข้อมูลสัญญาณเอ็นเอ็มอาร์โดยเฉพาะ ซึ่งข้อมูลนี้จะถูกเก็บไว้ในรูปของเมทริกซ์ที่มีขนาด 32x32 และในการทำงานใดๆ ที่เกี่ยวกับข้อมูลดังกล่าว เช่นการวิเคราะห์ด้วยเอฟเอฟที ก็จะทำงานบนเมทริกซ์นี้ หรือเมื่อกล่าวถึงการอ่านหรือบันทึกข้อมูลก็จะหมายถึงข้อมูลที่อยู่ในเมทริกซ์นี้เอง

เมื่อเตรียมเนื้อที่และค่าเริ่มต้นในตัวแปรต่างๆเรียบร้อยแล้ว โปรแกรมก็จะแสดงข้อมูลเกี่ยวกับโปรแกรมและหน่วยความจำที่ใช้ให้ดูก่อนดังในภาพที่ ก.1



ภาพที่ ก.1

และเมื่อกดปุ่มใดๆอีกครั้งหนึ่งโปรแกรมก็จะแสดงให้เห็นว่ามีหน่วยความจำเหลืออยู่ให้ใช้งานได้อีกเท่าใด พร้อมกับแสดงเมนูหลักให้ดูตามแนวนอนดังในภาพที่ ก.2



ภาพที่ ก.2

ซึ่งจะมีหัวข้อใหญ่ให้เลือกใช้งานได้ 6 หัวข้อด้วยกัน คือ

1.dAta

2.Make up

3.FFT

4.Display

5.Options

6.exit

ซึ่งในแต่ละหัวข้อก็จะมีหัวข้อย่อยลงไปให้เลือกใช้งานได้อีก โดยการเลือกใช้หัวข้อนี้สามารถทำได้สองวิธีคือ

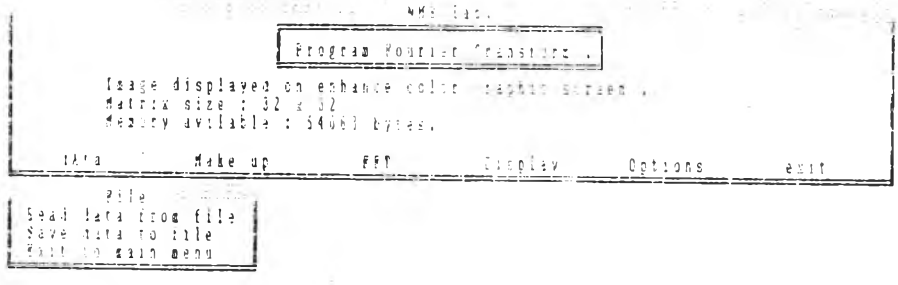
1.กดปุ่มตัวอักษรที่เป็นตัวใหญ่ของหัวข้อนั้นๆ เช่น ตัว 'A' ในหัวข้อ dAta

2.กดปุ่มลูกศรเพื่อเลื่อนแถบสว่างที่อยู่บนเมนูไปยังหัวข้อที่ต้องการแล้วกดปุ่ม Enter

เมื่อเลือกหัวข้อใหญ่แล้วจะพบว่าในแต่ละหัวข้อใหญ่จะมีหัวข้อย่อยปรากฏขึ้นซึ่งขึ้นกับว่าหัวข้อใหญ่

นั้นเป็นเรื่องเกี่ยวกับอะไรดังรายละเอียดที่จะแสดงให้ดูต่อไปนี้

1. Data หัวข้อนี้เป็นเรื่องเกี่ยวกับการอ่านและบันทึกข้อมูลที่จะมาวิเคราะห์ เมื่อเลือกหัวข้อนี้จะมีหัวข้อย่อยมาปรากฏให้เห็นดั่งในภาพที่ ก.3



ภาพที่ ก.3

ซึ่งแต่ละหัวข้อย่อยจะมีหน้าที่ดังนี้

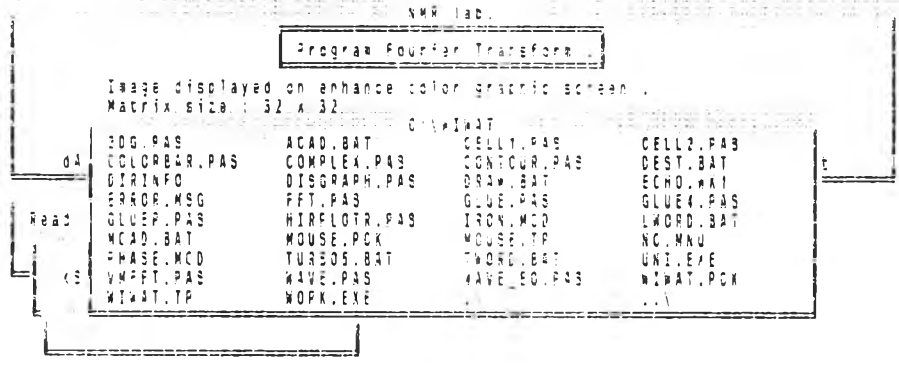
1.1 Read data from file รายการนี้จะทำหน้าที่อ่านข้อมูลจากแฟ้มข้อมูลในแผ่นดิสค์เข้ามาเก็บไว้ในหน่วยความจำ

1.2 Save data to file รายการนี้จะทำหน้าที่บันทึกข้อมูลจากหน่วยความจำลงในแฟ้มข้อมูลในแผ่นดิสค์

1.3 Exit to main menu กลับสู่เมนูหลัก

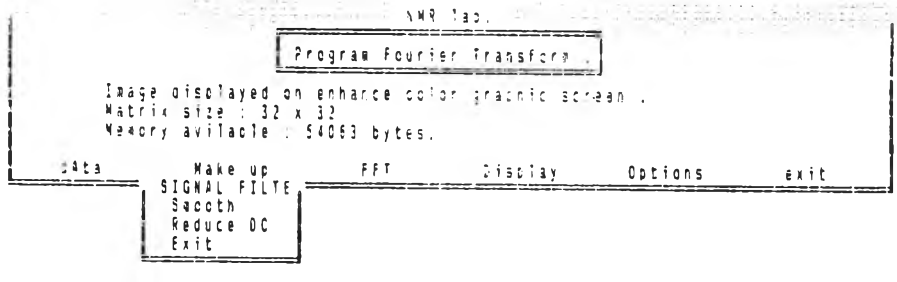
ในการอ่านหรือบันทึกข้อมูลบนแผ่นดิสค์นี้ โปรแกรมจะถามชื่อของแฟ้มข้อมูลที่จะอ่านหรือบันทึกข้อมูล ในกรณีที่ต้องการดูรายชื่อแฟ้มข้อมูลที่อยู่บนแผ่นดิสค์ก็สามารถทำได้โดยกด

ปุ่มเว้นวรรค 1 ครั้งแล้วกดปุ่ม Enter โปรแกรมจะแสดงชื่อของแฟ้มข้อมูลที่มีอยู่ขึ้นมาให้ดูดังภาพที่ ก.4 ซึ่งจะสามารถเลือกแฟ้มข้อมูลได้โดยใช้ปุ่มกดลูกศรเลื่อนแถบสว่าง ไปยังชื่อแฟ้มที่ต้องการแล้วกดปุ่ม Enter



ภาพที่ ก.4

2. Make up หัวข้อนี้จะใช้ในการปรับปรุงข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำ ซึ่งทำได้สองแบบคือ (ดูภาพที่ ก.5)



ภาพที่ ก.5

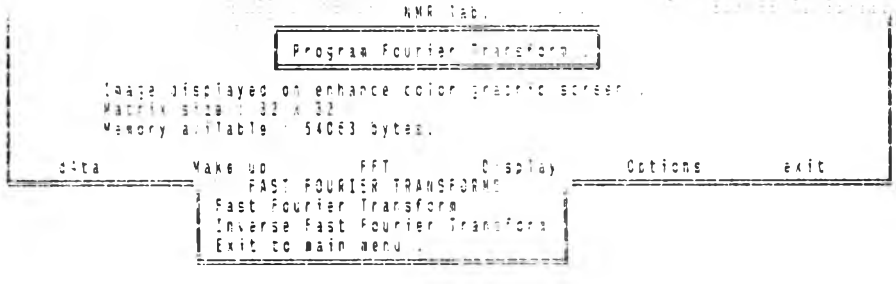
2.1 Smooth จะทำหน้าที่ลดสัญญาณรบกวนที่มีความถี่สูงออกจากสัญญาณจริง โดยการปรับรูปกราฟที่ไม่เรียบให้ราบเรียบมากขึ้นและนอกจากการลดสัญญาณรบกวนแล้วยังสามารถใช้ในการตกแต่งรูปภาพที่ได้ออกมาได้อีกด้วย

2.2 Reduce DC ทำหน้าที่ลดสัญญาณระดับศูนย์ซึ่งอาจมีปะปนมาเนื่องจากเครื่องมือที่ไม่สมบูรณ์ ซึ่งจะทำให้รูปภาพที่ได้มีลักษณะผิดไปจากที่ควรจะเป็น

อย่างไรก็ดีการตกแต่งหรือปรับปรุงข้อมูลนี้ควรใช้ในกรณีที่มีความจำเป็นเพียงเท่านั้นเนื่องจากการปรับปรุงข้อมูล ไม่ว่าจะด้วยวิธีใด เพื่อกำจัดส่วนที่ไม่ต้องการออกไปย่อมมีผลกระทบกับข้อมูลส่วนที่ต้องการเสมอโดยเฉพาะในการวิเคราะห์ข้อมูลโดยใช้การแปลงแบบฟูเรียร์นี้ ข้อมูลทั้งหมดจะมีความสัมพันธ์กับทุกจุดบนภาพที่ได้ออกมาจากการแปลง

3. FFT เรื่องเอฟเอฟทีหรือการแปลงแบบฟูเรียร์อย่างเร็วนี้ถือเป็นส่วนที่สำคัญที่สุดของโปรแกรม แบ่งออกเป็น 2 ประเภทด้วยกันคือ (ภาพที่ ก.6)

3.1 Fast Fourier Transform ใช้สำหรับการแปลงข้อมูลที่อยู่ในเมทริกซ์ภายในหน่วยความจำไปโดยวิธีการแปลงแบบฟูเรียร์อย่างเร็วในสองมิติ ซึ่งเมื่อทำการแปลงเรียบร้อยแล้วจะนำผลการแปลงไปเก็บไว้ในเมทริกซ์เดียวกันนั้นเอง



ภาพที่ ก.6

3.2 Inverse Fast Fourier Transform เป็นการแปลงข้อมูลในเมทริกซ์ โดยวิธีการแปลงผกผันแบบฟูเรียร์อย่างเร็วซึ่งวิธีนี้เป็นการกระทำที่ผกผันกับการกระทำในข้อ 3.1 ดังนั้นถ้าหากใช้การแปลงแบบข้อ 3.2 ลงไปบนผลของการแปลงแบบข้อ 3.1 จะได้ผลเป็นเมทริกซ์ เดิมที่ยังไม่ผ่านการแปลง

4. Display นับเป็นส่วนที่มีความสำคัญรองจากหัวข้อที่ 3 เลยที่เดียว เนื่องจากวัตถุประสงค์ของการทดลองนี้ก็คือ การนำข้อมูลมาวิเคราะห์และแสดงผลในเชิงรูปภาพซึ่งโปรแกรมนี้ก็ได้ตอบสนองความต้องการนี้ได้เป็นอย่างดี โดยสามารถนำข้อมูลในเมทริกซ์ในหน่วยความจำแสดงผลเป็นภาพให้ดูได้ในหลายรูปแบบ ตามแต่ความต้องการและวัตถุประสงค์ของผู้ใช้ว่าต้องการพิจารณาข้อมูลในลักษณะใด อย่างไรก็ดีการแสดงภาพก็พอจะแยกออกได้เป็นสองรูปแบบใหญ่ๆคือ

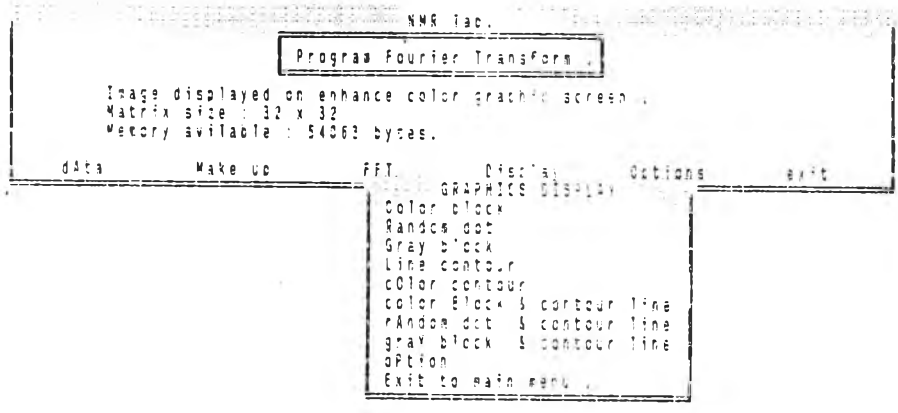
1. ในลักษณะเป็นค่าสัมบูรณ์ของแต่ละจุดข้อมูล

2. ในลักษณะเป็นเส้นชั้นของความสูงหรือคอนทัวร์ (Contour)

การแสดงภาพในลักษณะที่เป็นค่าสัมบูรณ์ของจุดข้อมูล จะเป็นการนำค่าของจุดข้อมูลในแต่ละจุดซึ่งเป็นค่าเชิงซ้อนมาหาค่าสัมบูรณ์แล้วนำมาเขียนลงบนจอภาพ โดยอาจจะใช้สี ขนาด หรือความหนาแน่นของจุดแสดงแทนขนาดของค่าสัมบูรณ์ว่ามีค่าโดยประมาณเป็นเท่าใด

ส่วนการแสดงภาพในลักษณะเป็นเส้นชั้นของความสูง จะนำข้อมูลทั้งหมดมาทำการประมาณค่าว่าในตำแหน่งที่ไม่ใช่ตำแหน่งที่มีข้อมูลอยู่ควรจะมีค่าประมาณเท่าใด แล้วลากเส้นไปตามแนวที่มีค่าระดับความสูงของข้อมูลเท่ากัน วิธีนี้จะให้ภาพที่ดูดีกว่าและต่อเนื่องกันไปตลอดทั้งภาพ

หัวข้อย่อยที่สามารถเลือกได้ในเรื่องนี้มีอยู่หลายหัวข้อย่อยด้วยกัน ดังแสดงในภาพที่ 7.7 ซึ่งแต่ละหัวข้อจะแสดงภาพโดยวิธีต่างๆ กันดังนี้คือ



ภาพที่ ก.7

4.1 Color block จะแสดงภาพเป็นจุดสี่เหลี่ยม โดยแต่ละจุดจะแสดงแทนข้อมูลในเมทริกซ์หนึ่งข้อมูล โดยจะใช้สีแต่ละสีแทนระดับขนาดสัมบูรณ์ของข้อมูลนั้นๆ

4.2 Random dot จะใช้ความหนาแน่นของจุดเล็กๆ แสดงแทนขนาดสัมบูรณ์ของ ข้อมูลที่ตำแหน่งนั้นๆ

4.3 Gray dot ใช้ขนาดของจุดสี่เหลี่ยมขนาดใหญ่แสดงแทนขนาดสัมบูรณ์ของข้อมูลที่ตำแหน่งนั้น

4.4 line Contour เป็นการแสดงเส้นชั้นของความสูงของขนาดสัมบูรณ์ของข้อมูลในเมทริกซ์ โดยสามารถกำหนดจำนวนเส้นแสดงระดับชั้นได้โดยใช้หัวข้อ oPtion

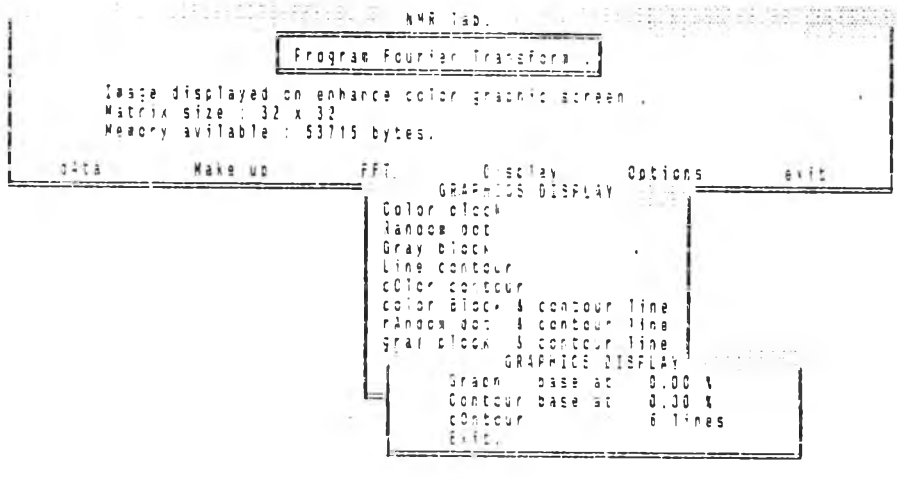
4.5 cOlor contour เป็นการแสดงระดับชั้นความสูงคล้ายกันกับข้อ 4.4 แต่จะใช้สีแสดงแทนช่วงระดับชั้นความสูงต่างๆ วิธีนี้ไม่สามารถเลือกจำนวนของระดับชั้นได้แต่จำนวนระดับชั้นจะถูกกำหนดโดยจำนวนสีที่มากที่สุดที่ระบบคอมพิวเตอร์ที่ใช้งานจะทำได้

4.6 color Block & contour line จะแสดงภาพทั้งแบบในข้อ 4.1 และข้อ 4.4 รวมกัน

4.7 rAndom dot & contour line จะแสดงภาพทั้งแบบในข้อ 4.3 และข้อ 4.4 รวมกัน

4.8 graY block & contour line จะแสดงภาพทั้งแบบในข้อ 4.3 และข้อ 4.4 รวมกัน

4.9 oPtion เป็นหัวข้อที่ใช้กำหนดรายการเป็นพิเศษในการเขียนภาพซึ่งจะมีรายการย่อยดังแสดงในภาพที่ ก.8



ภาพที่ ก.8

4.9.1 Graph base จะใช้กำหนดระดับต่ำสุดของข้อมูลที่จะนำมาใช้สร้างกราฟเพื่อแสดงค่าแอมพลิจูดของแต่ละจุดข้อมูล

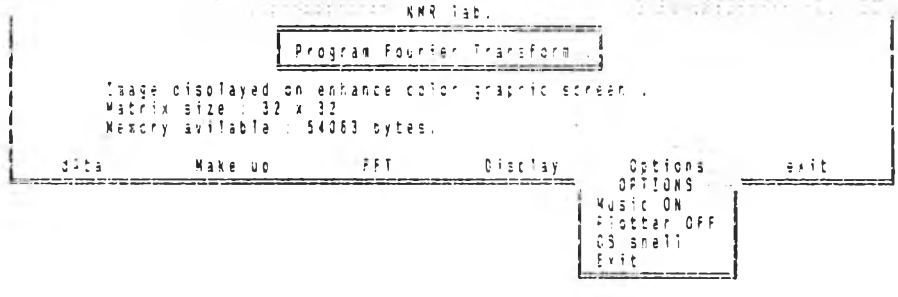
4.9.2 Contour base จะกำหนดระดับของเส้นแสดงระดับชั้นที่

แสดงระดับชั้นต่ำสุด ว่าจะใช้ระดับใดเป็นระดับต่ำสุด

4.9.3 cOntour line ใช้กำหนดจำนวนเส้นแสดงระดับชั้นความสูงที่จะวาดออกมา

สาเหตุที่ต้องเขียนโปรแกรมให้สามารถกำหนดระดับต่ำสุดที่จะนำมาแสดงได้ก็เนื่องจากว่าในการทดลอง มักจะมีสัญญาณรบกวนเข้ามาในระดับต่ำๆเสมอ ซึ่งถ้าหากปล่อยให้นำมาสร้างเป็นภาพออกแสดงทั้งหมดจะทำให้ภาพที่ได้ดูไม่สื่อความหมายเท่าที่ควร จึงได้ตัดสัญญาณส่วนที่มีค่าอัมปลิจูดต่ำออกไปจากการแสดงผล โดยกำหนดเป็นเปอร์เซนต์ของความสูงที่มากที่สุด

5. Option ใช้เลือกการทำงานต่างๆ ดังแสดงในภาพที่ ก.9



ภาพที่ ก.9

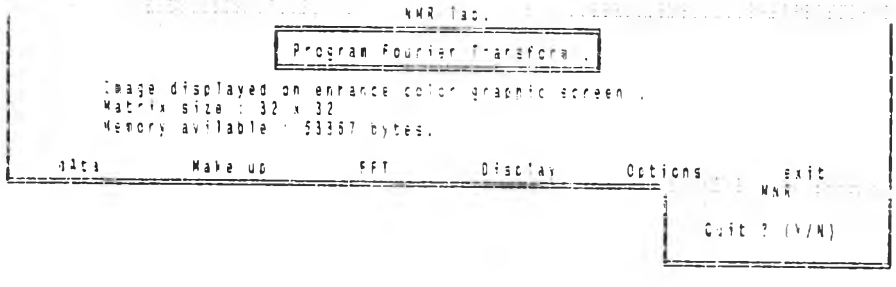
5.1 Music ON/OFF ใช้ควบคุมว่าเมื่อทำงานใดงานหนึ่งเสร็จเรียบร้อยแล้วจะให้ส่งเสียงเตือนหรือไม่ เนื่องจากว่าการทำงานบางอย่างอาจจะช้ามาก โดยเฉพาะถ้าใช้คอมพิวเตอร์ที่มีความเร็วในการทำงานต่ำ จึงได้ให้คอมพิวเตอร์ส่งเสียงเตือนผู้ใช้ได้

5.2 Plotter ON/OFF ใช้เลือกว่าจะให้แสดงภาพออกทางจอภาพหรือ

เขียนภาพออกทางเครื่องพลอตเตอร์ (Plotter) ในการใช้งานหัวข้อนี้ผู้ใช้จะต้องมีเครื่องพลอตเตอร์ต่ออยู่กับคอมพิวเตอร์ผ่านพอร์ต (Port) ชื่อ LPT2 โดยพลอตเตอร์นี้จะต้องเป็นพลอตเตอร์ชนิดที่ใช้คำสั่งแบบมาตรฐานของบริษัทฮิวเลตต์แพกการ์ด (Hewlett-packard) และเนื่องจากว่าการทำงานของพลอตเตอร์นี้จะช้าและเสียเวลามาก จึงได้มีระบบที่จะทำให้พลอตเตอร์ทำงานไประหว่างที่ใช้คอมพิวเตอร์ทำงานอื่นๆด้วย (เรียกว่า background print หรือ spooling) ซึ่งถ้าผู้ใช้เลือกใช้วิธีนี้จะต้องมีโปรแกรม Print.exe (ซึ่งเป็นโปรแกรมที่มีมากับระบบ MS-DOS) อยู่ในแผ่นดิสก์ที่ใช้งานอยู่ในขณะนั้นด้วย

5.3 OS shell เป็นการออกจากโปรแกรมไปชั่วคราว เพื่อไปใช้คำสั่งของ MS-DOS โดยที่ตัวโปรแกรมเองก็ยังคงอยู่ในหน่วยความจำ และสามารถกลับเข้าสู่โปรแกรมโดยใช้คำสั่ง EXIT

6. Exit เป็นการหยุดการทำงานของโปรแกรมและส่งการทำงานคืนให้แก่ MS-DOS โดยจะมีการถามยืนยันอีกครั้งหนึ่งว่าจะหยุดการทำงานแน่นอน หรือไม่ดังแสดงในภาพที่ ก.10



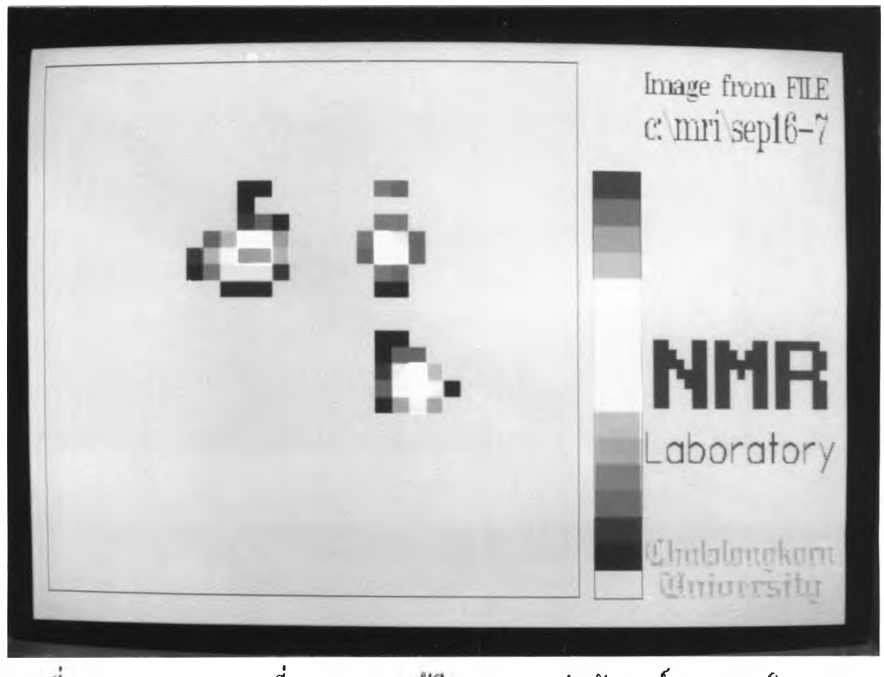
ภาพที่ ก.10

เนื่องจากโปรแกรมที่เขียนขึ้นมานี้ไม่ใช่โปรแกรมที่จะทำงานได้โดยอัตโนมัติทั้งหมด ดัง

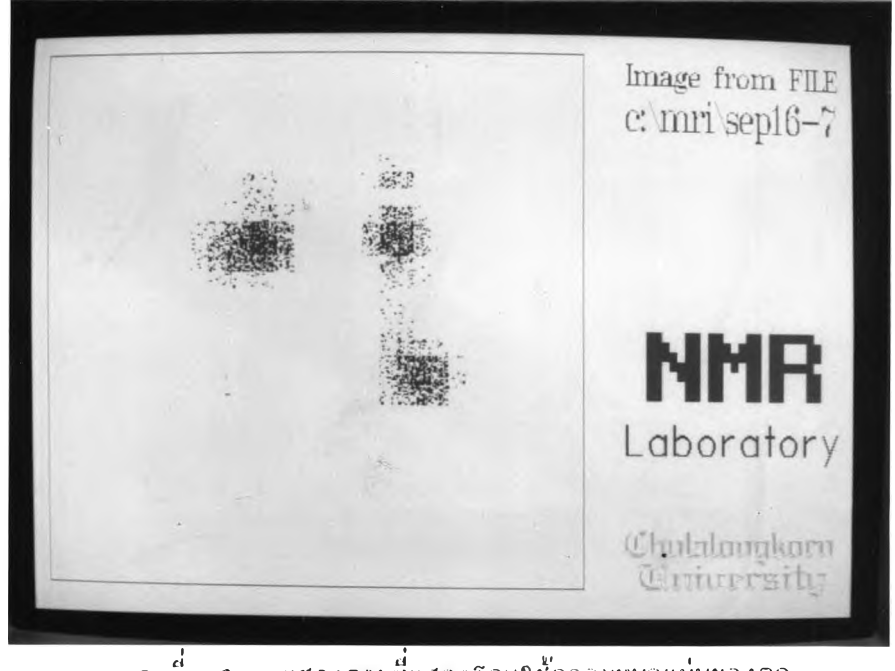
ดังนั้นจึงจำเป็นที่ผู้ใช้จะต้องมีความรู้ในการพัฒนาและวิธีการเรียกใช้งานโปรแกรม ตลอดจนถึงแนวคิดของการสร้างภาพจากวิธีเอ็นเอ็มอาร์อยู่พอสมควร จึงจะใช้งานได้อย่างมีประสิทธิภาพสูงสุด

ภาคผนวก ข

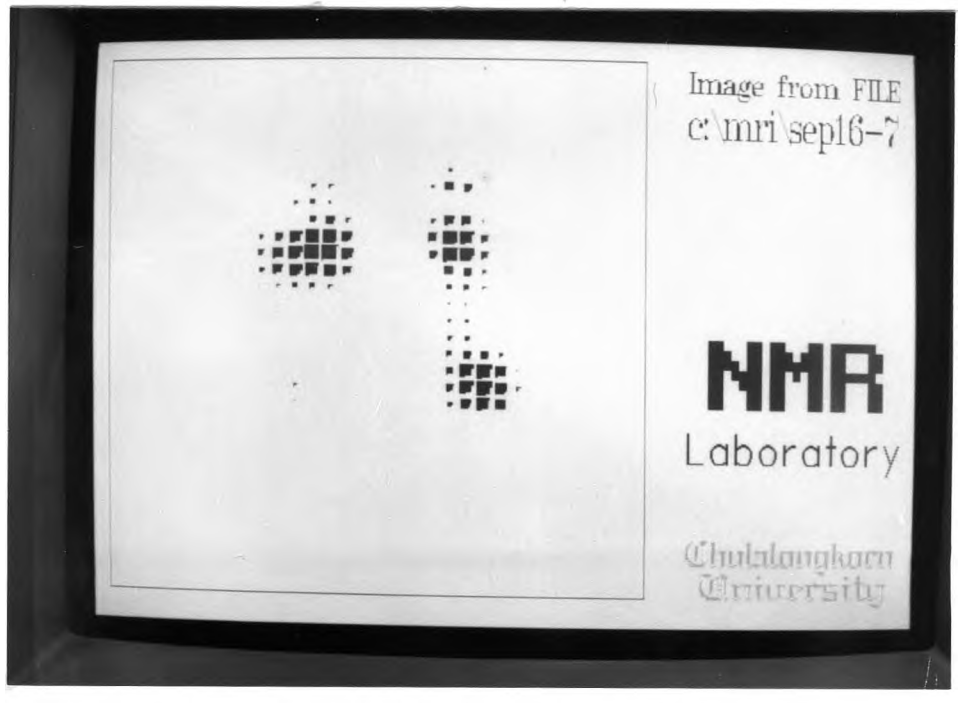
ภาพที่ได้จากการประมวลผล



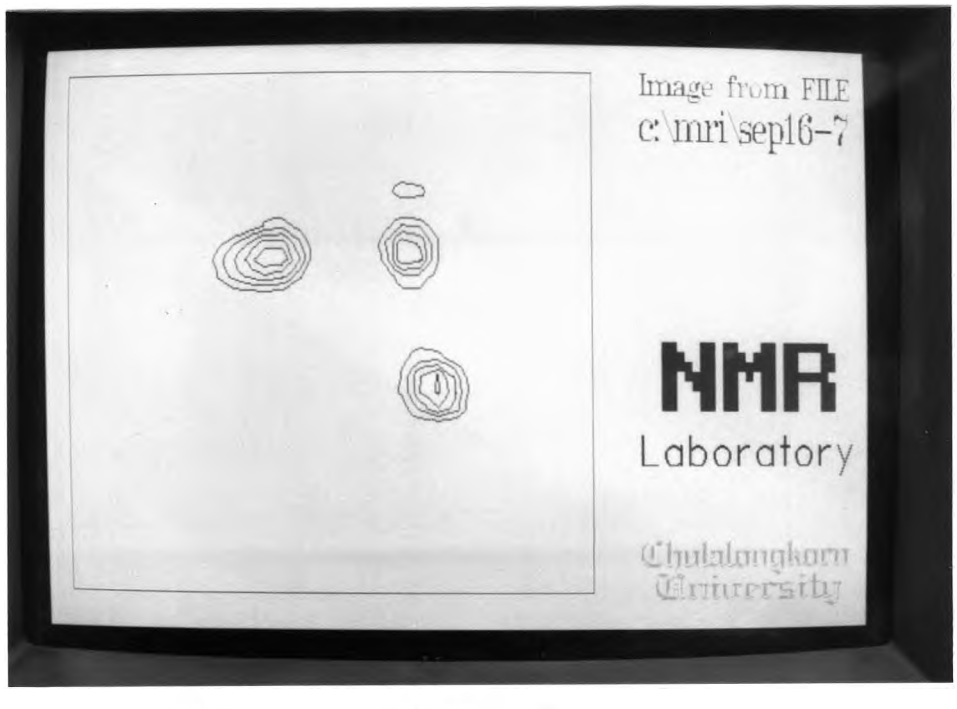
ภาพที่ 1 แสดงภาพที่แสดงโดยใช้สีแสดงแทนค่าสัมบูรณ์ของสมาชิกของเมทริกซ์



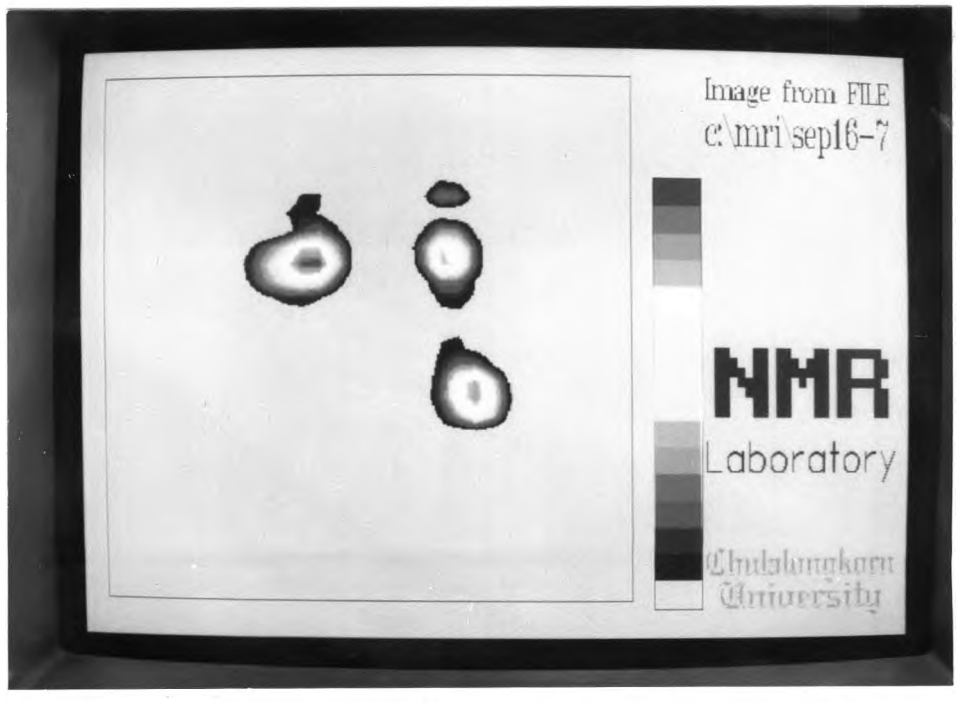
ภาพที่ 2 แสดงภาพที่แสดงโดยใช้ความหนาแน่นของจุด
แสดงแทนค่าสัมบูรณ์ของสมาชิกของเมทริกซ์



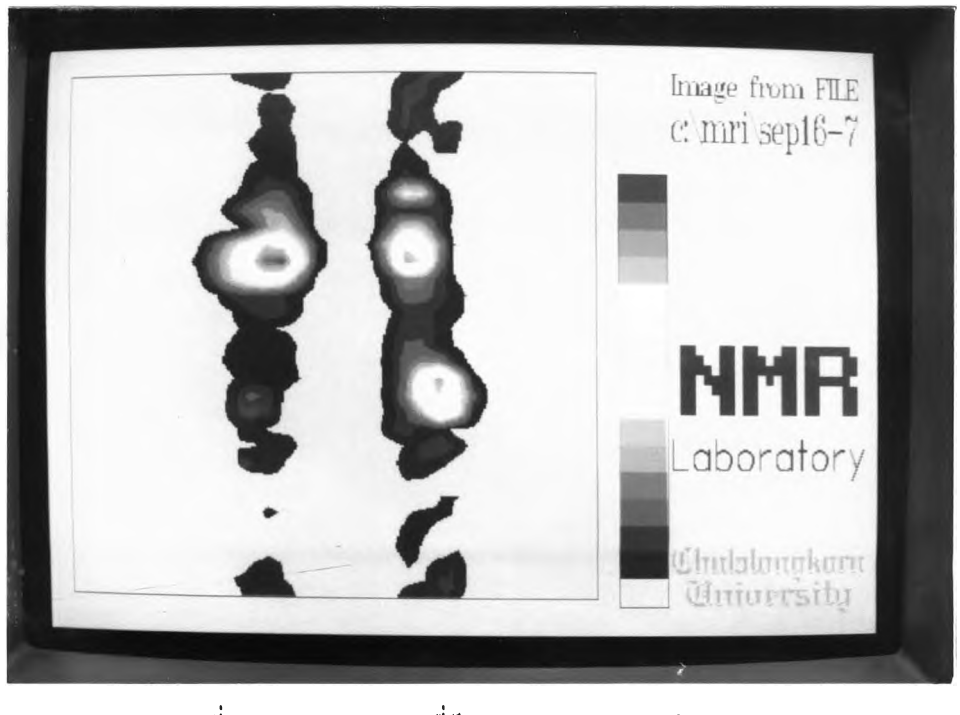
ภาพที่ 3 แสดงภาพที่แสดงโดยใช้ขนาดของจุดแสดงแทนค่าสัมบูรณ์ของสมาชิกของเมทริกซ์



ภาพที่ 4 แสดงภาพที่ใช้เส้นระดับความสูงแสดงแทนค่าโดยประมาณของสมาชิกของเมทริกซ์



ภาพที่ 5 แสดงภาพที่สีต่างๆแสดงแทนระดับความสูง
โดยประมาณของสมาชิกของเมทริกซ์



ภาพที่ 6 แสดงภาพที่สีต่างๆแสดงแทนระดับความสูงโดย
ประมาณของสมาชิกของเมทริกซ์ ซึ่งในภาพนี้
เป็นสัญญาณที่ยังไม่ได้ตัดสัญญาณรบกวนออก

ภาพที่ 7 แสดงการจัดวางหลอดทดลองที่ใช้ในการทดลอง

ภาคผนวก ค

วิธีการทำการแปลงแบบฟูเรียร์อย่างเร็ว

ในบทนี้จะได้แสดงการพิสูจน์ว่าวิธีการแปลงแบบฟูเรียร์อย่างไม่ต่อเนื่อง สามารถพัฒนากระบวนการให้สามารถทำการคำนวณได้เร็วขึ้น ซึ่งก็จะกลายเป็นกระบวนการที่ใช้ในการแปลงแบบฟูเรียร์อย่างเร็วนั่นเอง ดังจะได้แสดงให้เห็นดังต่อไปนี้

ถ้าให้ g_k เป็น ลำดับของจำนวนเชิงซ้อน

โดยที่ $k = 0,1,.....,N-1$

จะเขียนได้ว่า

$$G_j = \sum_{k=0}^{N-1} g_k \cdot e^{(2\pi i)\cdot jk/N} \quad ; \quad i = -1$$

เป็น DFT ของ g_k

$$= \sum_{k=0}^{N-1} g_k \cdot w^{jk}$$

เมื่อ $w = e^{2\pi i/N}$

ซึ่งอนุมานได้ทั่ว

$$G(x) = \int_{-\infty}^{\infty} g(t) \cdot e^{2\pi i x t} \cdot dt$$

ให้

$$N = 2 \times 2 \times 2 \times .....\times 2 = 2^t$$

เราสามารถเขียน j ได้ในรูปของ

$$j = j_1 + 2 \cdot j_2 + 2^2 j_3 + 2^3 j_4 + \ldots\ldots + 2^{t-1} j_t$$

โดยที่ $j_1, j_2, j_3, \ldots, j_t = 0, 1$

และ

$$k = k_t + 2 \cdot k_{t-1} + 2^2 k_{t-2} + \ldots\ldots\ldots + 2^{t-1} k_1$$

เมื่อ $k_1, k_2, k_3, \ldots, k_t = 0, 1$

หรือเขียนอย่างสั้นๆได้เป็น

$$j = \sum_{s=1}^{t} j_s 2^{s-1}$$

$$k = \sum_{s=1}^{t} k_s 2^{t-s}$$

สามารถเขียนได้เป็น

$$w^{j \cdot k} = w^{j \cdot \left(k_t + 2\, k_{t-1} + 2^2 k_{t-2} \ldots\ldots + 2^{t-1} k_1\right)}$$

$$G_j = \sum_{k=0}^{N-1} g_k \cdot w^{j \cdot k}$$

$$= \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \; w^{j\cdot(k_t + 2\cdot k_{t-1} + 2^2 k_{t-2} + \cdots + 2^{t-1}k_1)}$$

$$= \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \cdot w^{j\cdot k_t} \cdot w^{j\cdot 2\cdot k_{t-1}} \cdot w^{j\cdot 2^2 k_{t-2}} \cdots w^{j\cdot 2^{t-1}k_1}$$

แทนค่า

$$j = j_1 + 2\cdot j_2 + 2^2 j_3 + 2^3 j_4 + \cdots + 2^{t-1} j_t$$

จะได้

$$= \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \cdot w^{\begin{matrix}(j_1+2\cdot j_2 + \cdots + 2^{t-1}j_t)\cdot k_t \\ (2\cdot j_1+2^2\cdot j_2 + \cdots + 2^t\cdot j_t)\cdot k_{t-1} \\ \cdots\cdots \\ (2^{t-1}\cdot j_1+2^t j_2 + \cdots + 2^{2t-2}j_t)\cdot k_1\end{matrix}}$$

$$G_j = \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \cdot \begin{bmatrix} j_1\cdot k_t & 2\cdot j_2 k_t & \cdots & 2^{t-1}j_t\cdot k_t \\ w & \cdot w & \cdots & w \end{bmatrix} \begin{bmatrix} 2\cdot j_1\cdot k_{t-1} & 2^2 j_2\cdot k_{t-1} & \cdots & 2^t j_t\cdot k_{t-1} \\ w & \cdot w & \cdots & w \end{bmatrix} \cdots\cdots \begin{bmatrix} 2^{t-1}\cdot j_1\cdot k_1 & 2^t\cdot j_2\cdot k_1 & \cdots & 2^{2t-2}\cdot j_t\cdot k_1 \\ w & \cdot w & \cdots & w \end{bmatrix}$$

แต่ว่า $w^{a\cdot N} = 1$ เมื่อ a เป็นจำนวนเต็มใดๆ

เมื่อ

$$2^t \quad 2^{t+1} \quad 2^{t+2} \qquad\qquad 2^{2t-1}$$

$$w = w = w = \ldots\ldots = w = 1$$

จะได้

$$G_j = \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \cdot \begin{bmatrix} j_1 k_t & 2 \cdot j_2 k_t & \cdots & 2^{t-1} j_t k_t \\ w & \cdot w & \cdots & w \end{bmatrix} \begin{bmatrix} 2 \cdot j_1 k_{t-1} & 2^2 j_2 k_{t-1} & \cdots & 2^{t-1} j_{t-1} k_{t-1} \\ w & \cdot w & \cdots & w \end{bmatrix} \cdots\cdots \begin{bmatrix} 2^{t-1} j_1 k_1 \\ w \end{bmatrix}$$

$$G_j = \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_t=0}^{1} g_k \cdot w^{(j_1 + 2 \cdot j_2 + \cdots + 2^{t-1} j_t) \cdot k_t} \; w^{(2 \cdot j_1 + 2^2 j_2 + \cdots + 2^{t-1} j_{t-1}) \cdot k_{t-1}} \cdots\cdots w^{2^{t-1} j_1 k_1}$$

$$G_j = \sum_{k_1=0}^{1} \cdots \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} g_k \cdot w^{2^{t-1} j_1 k_1} \; \cdots\cdots \; w^{(2 \cdot j_1 + 2^2 j_2 + \cdots + 2^{t-1} j_{t-1}) \cdot k_{t-1}} \; w^{(j_1 + 2 \cdot j_2 + \cdots + 2^{t-1} j_t) \, k_t}$$

จัดลำดับของวงเล็บใหม่ โดยแยกเทอมที่ไม่มี summation index ออกไปอยู่นอกวงเล็บ

$$G_j = \sum_{k_t=0}^{1}\left[\sum_{k_{t-1}=0}^{1}\cdots\left[\sum_{k_2=0}^{1}\left[\sum_{k_1=0}^{1} g_k \cdot w^{2^{t-1}\cdot j_1\cdot k_1}\right] w^{2^{t-2}\cdot(j_1+2\cdot j_2)\cdot k_2}\right]\right]$$

$$\cdots\cdots$$

$$w^{(2\cdot j_1 + 2^2 j_2 + \cdots + 2^{t-1} j_{t-1})\cdot k_{t-1}}$$

$$w^{(j_1 + 2\cdot j_2 + \cdots + 2^{t-1} j_t)\cdot k_t}$$

จากสมการสุดท้ายนี้จะเห็นได้ว่า ถ้าเราเริ่มคำนวณจากวงเล็บในสุดแล้วเก็บค่าไว้แล้วเลื่อนมาคำนวณชั้นนอกๆถัดมาเรื่อยๆ โดยแปรค่าของ $j_1, j_2, \ldots, j_t$ ไปเรื่อยๆจนครบทุกค่าที่เป็นไปได้แล้ว เราก็จะได้ค่าของ G_j ครบทุกตัวตามไปด้วย เมื่อคำนวณไปจนถึงวงเล็บนอกสุด

อย่างไรก็ดี สมการและวิธีการที่แสดงไว้นี้ก็ยังไม่เหมาะกับการนำไปใช้ในการเขียนโปรแกรมคอมพิวเตอร์โดยตรง จึงต้องพัฒนารูปแบบต่อไปอีกเล็กน้อย จากสมการและวิธีการข้างต้นนี้

เริ่มต้นที่กำหนดตัวแปร $f_1(k_1, k_2, \ldots, k_t)$ ขึ้นมา ; $l = 1,2,\ldots,t$

$k_s = 0,1$

เมื่อ t มีค่ากำหนดโดย

$$N = 2^t$$

จากนั้นให้

$$f_0(k_1, k_2, k_3, \ldots, k_t) = g_k$$

เมื่อ

$$k = \sum_{s=1}^{t} k_s 2^{t-s} \quad ; \quad k_s = 0,1 \quad ; \quad s = 1,2,\ldots,t$$

จากนิยามที่

$$f_1(j_1,k_2,k_3,\ldots,k_t) = \sum_{k_1=0}^{1} f_0(k_1,k_2,k_3,\ldots,k_t)\cdot w^{2^{t-1}j_1k_1}$$

$$f_2(j_1,j_2,k_3,\ldots,k_t) = \sum_{k_2=0}^{1} f_1(j_1,k_2,k_3,\ldots,k_t)\cdot w^{2^{t-1}(j_1+2\cdot j_2)\cdot k_1}$$

$$f_3(j_1,j_2,j_3,\ldots,k_t) = \sum_{k_3=0}^{1} f_2(j_1,j_2,k_3,\ldots,k_t)\cdot w^{2^{t-1}(j_1+2\cdot j_2+2^2 j_3)\cdot k_1}$$

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

$$f_{t-1}(j_1,j_2,\ldots,j_{t-1},k_t) = f_2(j_1,\ldots j_{t-2},0,k_t) + f_2(j_1,\ldots j_{t-2},1,k_t)\cdot w^{2\cdot(j_1+2\cdot j_2+\ldots+2^{t-2}j_{t-1})}$$

$$f_t(j_1,j_2,\ldots,j_{t-1},j_t) = f_2(j_1,\ldots j_{t-2},j_{t-1},0) + f_2(j_1,\ldots j_{t-2},j_{t-1},1)\cdot w^{(j_1+2\cdot j_2+\ldots+2^{t-1}j_t)}$$

เมื่อ $j_s = 0,1 \quad ; s = 1,2,\ldots,t$ ตามนิยามของ

ซึ่งในที่สุดเราจะได้

$$G_j = f_t(j_1, j_2, \ldots, j_t)$$

เมื่อ

$$j = \sum_{s=1}^{t} j_s \cdot 2^{s-1} \quad ; \quad j_s = 0,1 \quad ; \quad s = 1,2,\ldots,t$$

ซึ่งกระบวนการทั้งหมดนี้ เราอาจเขียนให้สั้นๆได้ดังนี้

ขั้นแรก ให้

$$f_0(k_1, k_2, k_3, \ldots, k_t) = g_k \qquad \ldots\ldots\ldots(1)$$

จากนั้น ทำซ้ำ

$$f_l(j_1, j_2, \ldots, j_l, k_{l+1}, \ldots, k_t) = \sum_{k=0}^{l} f_{l-1}(j_1, \ldots, j_{l-1}, k, \ldots, k_t) \cdot w^{2^{l-1} \cdot \left(\sum_{s=1}^{l} j_s 2^{l-1}\right) \cdot k_t} \qquad \ldots\ldots(2)$$

โดยที่ค่า $l = 1,2,3,\ldots,t$

และในที่สุด ให้

$$G_j = f_t(j_1, j_2, \ldots, j_t) \qquad \ldots\ldots\ldots(3)$$

ถึงแม้ว่ากระบวนการที่ได้แสดงให้เห็นข้างต้นนี้ จะมีความเหมาะสมในการใช้งานในด้านการเขียนโปรแกรมคอมพิวเตอร์แล้วก็ตามแต่ทว่า จากสมการที่แสดงจะเห็นได้ว่า ถ้าหากมีการเปลี่ยนแปลงจำนวนข้อมูลที่จะใช้ในการคำนวณแล้ว จะต้องมีการเปลี่ยนจำนวนมิติของตัวแปรชนิดแถวลำดับ (array variable) ไปด้วย ซึ่งจะเป็นการยุ่งยากอย่างมาก ดังนั้น จึงได้มีการพัฒนาต่อไปอีกเล็กน้อยเพื่อการจัดข้อมูลพร่องนี้ออกไป ดังนี้

กำหนดตัวแปรใหม่

$$J_1 = 0$$

$$J_l = j_1 + 2 \cdot j_2 + \ldots + 2^{l-2} j_{l-1} \quad ; \quad l = 2, \ldots, t \quad ; \quad j_s = 0,1$$

$$= \sum_{s=1}^{l-1} 2^{s-1} j_s$$

และ

$$K_l = k_{l+1} + 2 \cdot k_{l+2} + \ldots + 2^{t-l-1} k_t \quad ; \quad l = 0, \ldots, t-1 \; ; \; k_s = 0,1$$

$$= \sum_{s=l+1}^{t} 2^{s-l-1} k_s$$

$$K_t = 0$$

นั่นคือ

$$J_l = 0,1,\ldots,2^{l-1}-1$$

$$K_l = 0,1,\ldots,2^{t-l}-1$$

และเราสามารถเขียนได้ว่า

$$j = J_l + 2^{l-1} j_l + 2^l K_l \quad ; \quad j_l = 0,1$$

โดยที่ $j = 0,1,2,\ldots,2^t-1$

เขียน f_l ใหม่เป็น

$$f_l(j_1, j_2, .., j_l, k_{l+1}, .., k_t) = f_l(j) = f_l(J_l + 2^{l-1} \cdot j_l + 2^l K_l)$$

นั่นคือเราสามารถเขียนสมการ (1), (2), และ (3) ข้างต้นให้อยู่ในรูปที่รวบรัดได้ดังนี้

เริ่มต้นให้

$$f_0(K_0) = g_k \quad ; \quad k = K_0$$

จากนั้นทำซ้ำ

$$f_l(J_l+2^{l-1}j_l+2^lK_l) = \sum_{k_l=0}^{1} f_{l-1}(J_l+2^{l-1}k_l+2^lK_l)\, w^{2^{t-1}J_{l+1}+k_l} \quad \ldots.(4)$$

เมื่อ

$$J_l = 0,1,\ldots,2^{l-1}-1 \quad ; \quad K_l = 0,1,\ldots,2^{t-l-1}-1 \quad ; \quad j_l = 0,1$$

$$l = 1,2,\ldots,t$$

ในที่สุดก็จะได้

$$G_j = f_t(J_{l+1}) \quad ; \quad j = J_{l+1}$$

เมื่อ

$$J_{l+1} = J_l + j_l 2^{l-1} \quad \ldots\ldots.(5)$$

พิจารณาในสองกรณี

กรณีที่ $j_0 = 0$

สมการ (4) จะกลายเป็น

$$f_l(J_l+2^lK_l) = f_{l-1}(J_l+2^lK_l) + f_{l-1}(J_l+2^{l-1}+2^lK_l)\cdot w^{2^{t-1}J_l}$$

กรณีที่ $j_0 = 1$

สมการ (4) จะกลายเป็น

$$f_l(J_l+2^{l-1}+2^l K_l) = f_{l-1}(J_l+2^l K_l) + f_{l-1}(J_l+2^{l-1}+2^l K_l)\cdot w^{2^{t-l}(J_{l+1}+2^{l-1})}$$

แต่

$$2^{t-l}2^{l-1} = 2^{t-1} = N/2$$

$$\Longrightarrow \quad w^{N/2} = -1$$

จะได้ว่า

$$f_l(J_l+2^{l-1}+2^l K_l) = f_{l-1}(J_l+2^l K_l) - f_{l-1}(J_l+2^{l-1}+2^l K_l)\cdot w^{2^{t-1}J_l}$$

ถ้าให้

$$p = J_l + 2^l \cdot K_l$$

และ

$$q = J_l + 2^l K_l + 2^{l-1} = p + 2^{l-1}$$

จะเขียนได้ว่า

<u>สำหรับกรณีที่ $j_0 = 0$</u>

$$f_l(p) = f_{l-1}(p) + f_{l-1}(q)\cdot w^{2^{t-l}J_l}$$

<u>สำหรับกรณีที่ $j_0 = 1$</u>

$$f_l(p) = f_{l-1}(p) - f_{l-1}(q)\cdot w^{2^{t-l}J_l}$$

ซึ่งจากทั้งหมดนี้จะเขียนเป็นกระบวนการทำงานได้ดังนี้คือ

ในการทำการแปลงแบบฟูเรียร์อย่างเร็วจากลำดับ $\{ g_k \}$ ไปยังลำดับ $\{ G_j \}$

ขั้นแรก ให้

$$f_0(K_0) = g_k \quad ; \quad k = K_0$$

จากนั้น ทำซ้ำสำหรับทุกค่า $l = 1,2,\ldots,t$

$$f_l(p) = f_{l-1}(p) + f_{l-1}(q) \cdot w^{2^{t-l} J_l}$$

$$f_l(p) = f_{l-1}(p) - f_{l-1}(q) \cdot w^{2^{t-l} J_l}$$

เมื่อ

$$p = J_l + 2^l \cdot K_l$$

และ

$$q = J_l + 2^l K_l + 2^{l-1} = p + 2^{l-1}$$

และค่า J_l , K_l แปรไปเรื่อยๆ

$$J_l = 0,1,\ldots,2_{l-1}-1$$

$$K_l = 0,1,\ldots,2^{t-l}-1$$

ในที่สุดจะได้

$$G_j = f_t(J_{t+1}) \quad ; \quad j = J_{l+1}$$

ภาคผนวก ง

ภาษาปาสคาลและโปรแกรมตัวแปลภาษาเทอร์โบปาสคาล

ในการวิเคราะห์ข้อมูลโดยใช้เครื่องคอมพิวเตอร์นี้ โปรแกรมที่ใช้งานถูกเขียนขึ้นด้วยภาษาปาสคาล และใช้โปรแกรมตัวแปลภาษาชื่อเทอร์โบปาสคาล รุ่น 5.0 (Turbo Pascal version 5.0) ซึ่งเป็นผลผลิตของบริษัท บอร์แลนด์อินเตอร์เนชั่นแนล (Borland International) แห่งสหรัฐอเมริกา

เหตุที่เลือกเขียนโปรแกรมด้วยภาษาภาษาปาสคาลนี้ก็เพราะว่า ภาษานี้เป็นภาษาที่สามารถเขียนได้อยู่ในลักษณะเป็นโครงสร้าง (structure program) ได้ง่าย ซึ่งการเขียนโปรแกรมเป็นโครงสร้างนี้จะทำให้การทำความเข้าใจโปรแกรมเป็นไปได้โดยสะดวก ทำให้การพัฒนาโปรแกรม ทั้งโดยผู้เขียนเองหรือผู้อื่นอาจจะต้องทำการพัฒนาโปรแกรมนี้ต่อไปทำได้ง่ายขึ้นและโอกาสที่จะเกิดข้อผิดพลาดในโปรแกรมลดน้อยลง

และการที่เลือกใช้โปรแกรมตัวแปลภาษาเทอร์โบปาสคาลนี้ก็ด้วยเหตุว่า เป็นโปรแกรมที่มีประสิทธิภาพสูงในการแปล กล่าวคือ มีความเร็วในการแปลสูง อนุญาติให้ใช้คำสั่งในการวาดภาพที่มีประสิทธิภาพ และสามารถแยกเขียนโปรแกรมเป็นส่วนๆแล้วนำมารวมกันในภายหลังได้

ภาษาปาสคาล

ภาษาปาสคาลเป็นภาษาคอมพิวเตอร์ระดับสูง ซึ่งสามารถนำไปเขียนโปรแกรมใช้งานได้อย่างทั่วไป ภาษานี้ออกแบบขึ้นครั้งแรกโดย ศาสตราจารย์ นิคคลัส ไวรท์ (Professor Niklaus Wirth) แห่งมหาวิทยาลัยเทคนิคแห่งซูริค (Technical University of Zurich) ประเทศสวิทเซอร์แลนด์ ส่วนชื่อว่าปาสคาลของภาษานี้

ศาสตราจารย์ ไวรท์ ตั้งขึ้นเพื่อเป็นเกียรติแก่ เบลส ปาสคาล (Blaise Pascal) ซึ่งเป็นนักคณิตศาสตร์และนักปรัชญาชาวฝรั่งเศสที่มีชีวิตอยู่ในศตวรรษที่สิบเจ็ด

ในครั้งแรกที่ศาสตราจารย์ ไวรท์ ออกแบบภาษานี้ขึ้นมาในปี พ.ศ.2514 นั้น ท่านได้มีเจตนารมณ์ที่จะใช้ภาษานี้ ในการสอนการโปรแกรมคอมพิวเตอร์อย่างเป็นระบบและวิธีการเขียนโปรแกรมเป็นโครงสร้าง แต่อย่างไรก็ดี ต่อมาการใช้งานภาษาปาสคาลนี้ได้แพร่หลายออกไปยังระบบคอมพิวเตอร์ต่างๆ และในปัจจุบันก็ได้กลายเป็นภาษาที่อยู่ในระดับแนวหน้าในบรรดาภาษาระดับสูงทั้งหลาย โดยมีการนำไปประยุกต์ใช้ทั้งในด้านการศึกษา การละเล่น และกระทั่งในการเขียนโปรแกรมใช้งานในระดับสูง

ลักษณะเด่นที่มีอยู่ในภาษาปาสคาลนี้ได้แก่ ความสามารถในการเขียนโปรแกรมเป็นโครงสร้าง และความสามารถที่จะสร้างตัวดำเนินการใหม่ได้ ความสามารถทั้งสองนี้มีประโยชน์มาก โดยเฉพาะในการเขียนโปรแกรมขนาดใหญ่ที่มีความยาวมากๆ เพราะจะทำให้ค่อยๆเขียนและทำการทดสอบ โปรแกรมไปทีละส่วนๆได้ ซึ่งจะเป็นผลให้การพัฒนาโปรแกรมได้ง่ายขึ้นและโอกาสที่โปรแกรมจะผิดพลาดจะมีน้อยลง

ตัวดำเนินการสร้างใหม่ได้ในภาษาปาสคาลนี้มีสองรูปแบบคือ

1. โพรซีเยอร์ (procedure)
2. ฟังชัน (function)

ซึ่งทั้งสองแบบนี้ แท้จริงแล้วก็คือโปรแกรมย่อยที่แยกออกมาเขียนไว้ต่างหาก และสามารถที่จะถูกเรียกใช้ได้จากส่วนอื่นๆของโปรแกรมนั้น และอาจเป็นโปรแกรมย่อยชนิดที่มีการเตรียมไว้ก่อนแล้ว หรืออาจเป็นชนิดที่ผู้เขียนโปรแกรมสร้างขึ้นเองก็ได้ ส่วนความแตกต่างของทั้งสองก็เป็นเพียงแต่ โพรซีเยอร์นั้นสามารถส่งค่ากลับได้โดยผ่านตัวผารามิเตอร์อย่างเดียว แต่สำหรับฟังชัน มันสามารถส่งค่ากลับด้วยชื่อของฟังชันได้อีกด้วย

ยูนิต

เนื่องจากหลักภาษาของภาษาปาสคาลนี้ อนุญาตให้สามารถใช้งาน ค่าคงที่ ประเภทข้อมูลตัวแปร โพรซีเยอร์ และฟังชัน ได้อย่างไม่จำกัดจำนวน ดังนั้น ในบางครั้งการเขียนโปรแกรมอาจเกิดการสับสนในการที่จะจัดสิ่งเหล่านี้ให้เป็นระเบียบได้โดยง่าย สำหรับเทอร์โบปาสคาลนี้ ได้มีเพิ่มเติมกระบวนการพิเศษจากภาษาปาสคาลมาตรฐาน คือ วิธีการใช้ยูนิต (unit) โดยที่โปรแกรมใดๆก็ตาม สามารถแบ่งออกเป็นส่วนๆตามความเหมาะสม แล้วนำไปแยกเก็บต่างหากจากกันได้ เราจะเรียกส่วนที่ถูกแยกออกไปนี้ว่าเป็น ยูนิต

ภายในยูนิตนี้อาจจะประกอบด้วย ค่าคงที่ ประเภทข้อมูล ตัวแปร โพรซีเยอร์ และฟังชัน ซึ่งจะคล้ายคลึงกันกับตัวโปรแกรมภาษาปาสคาลเอง เว้นแต่ว่า ส่วนโปรแกรมของยูนิต จะถูกเรียกให้ทำงานก่อนที่ตัวโปรแกรมหลักจะเริ่มทำงาน

อาจกล่าวได้ง่ายๆว่า ยูนิตนี้ แท้จริงแล้วก็คือคลังที่จะใช้สะสมสิ่งที่ผู้ใช้ต้องการเตรียมไว้ล่วงหน้าไว้ โดยที่ผู้ใช้อาจเลือกที่จะเรียกมาใช้หรือไม่ก็ได้ และเป็นสิ่งที่จะทำให้การเขียนและการแปลโปรแกรมสามารถแยกเป็นส่วนต่างหากจากกันได้

ยูนิตมีทั้งที่เตรียมไว้แล้วโดยเทอร์โบปาสคาล และทั้งที่ผู้ใช้โปรแกรมสามารถสร้างขึ้นเองได้ ยูนิตที่เตรียมไว้แล้วได้แก่ยูนิตที่ชื่อ System, Graph, Dos, Crt, และ Printer ทั้งหมดนี้จะถูกบันทึกอยู่ในแฟ้มข้อมูลชื่อ TURBO.TPL

การเรียกยูนิตใดยูนิตหนึ่งมาใช้งานนั้น ทำได้โดยใช้คำสั่ง uses เรียกชื่อของยูนิตนั้นๆก่อน แล้วจึงจะสามารถเรียกใช้งานสิ่งที่ถูกเตรียมไว้ในส่วนที่อยู่ระหว่างคำสั่ง interface และคำสั่ง implementation ของยูนิตนั้นได้

ภาคผนวก จ

ยูนิต analizer

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการปรับปรุงข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำ ซึ่งจะเรียกใช้โดยหัวข้อ Make up

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจดังนี้

1. โพรซิเยอร์ smoothing_matrix มีพารามิเตอร์เป็นเมทริกซ์ ใช้ในการลดสัญญาณรบกวนที่มีความถี่สูงออกจากสัญญาณจริงโดยปรับรูปร่างของข้อมูลในเมทริกซ์ที่ถูกส่งเข้ามาเป็นพารามิเตอร์นั้นให้เรียบขึ้น

2. โพรซิเยอร์ reduce_dc_from_matrix มีพารามิเตอร์เป็นเมทริกซ์ ใช้ในการลดสัญญาณกระแสตรงออกจาก เมทริกซ์ที่ถูกส่งเข้ามาเป็นพารามิเตอร์นั้น

3. โพรซิเยอร์ filtering_matrix มีพารามิเตอร์เป็นเมทริกซ์ เป็นโพรซิเยอร์ที่จะถูกเรียกเมื่อเลือกใช้หัวข้อ Make up ในขณะใช้งานโปรแกรม โพรซิเยอร์นี้จะทำหน้าที่เปิดหน้าต่างและเสนอเมนูขึ้นมาดังภาพที่ ก.5 เมื่อเมนูถูกเลือกก็จะเรียกใช้โพรซิเยอร์ smoothing_matrix หรือ reduce_dc_from_matrix ตามแต่กรณี

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไปแล้ว

```
unit analizer ;

interface

uses  crt     ,
      menu    ,
      cell    , { little tools }
      windsys,
      timing  ,
      complx  , {unit of complex variable and function }
      vardef  ; {variable definition unit}


procedure  smoothing_matrix ( var u : matrix ) ;

procedure reduce_dc_from_matrix ( var u : matrix ) ;

procedure ask_for_smoothing ( var u : matrix ) ;

procedure  ask_for_reduce_DC_from ( var u : matrix) ;

procedure  filtering_matrix ( var x : matrix ) ;


implementation


const  filter_menu : menu_type = nil ;
       filter_wind : window_type = nil ;


procedure  smoothing_matrix ( var u : matrix ) ;

var
    row         :  array [0..2] of vector ;
    i,j         ,
    first       ,
    second      ,
    third       :  integer ;

          procedure roll( var one,two,three : integer ) ;
          var temp : integer  ;
          begin
               temp := one   ; one   := two ;
               two  := three ; three := temp ;
          end ;

begin
     first := 0 ; second := 1 ; third := 2 ;

     row [first]  := u^[0]^ ;
     row [second] := u^[1]^ ;
```

```
      for  i :=  1 to max_data-2  do
      begin
        row [third] := u^[i+1]^ ;

            for  j :=  1 to max_data-2  do
            begin
              u^[i]^[j].re := row[first][j].re +
row[second][j+1].re +
                        row[second][j-1].re + row[third][j].re
+
                        4 * row[second][j].re ;
              u^[i]^[j].re := u^[i]^[j].re / 8.0 ;
              u^[i]^[j].im := row[first][j].im +
row[second][j+1].im +
                        row[second][j-1].im + row[third][j].im
+
                        4 * row[second][j].im ;
              u^[i]^[j].im := u^[i]^[j].im / 8.0 ;
            end {loop j} ;

            roll ( first , second , third ) ;

      end {loop i} ;
end {smooting_matrix} ;


procedure reduce_dc_from_matrix ( var u : matrix ) ;

var   sum   :  complex ;
      i,j   :  integer ;

begin

      for  i := 0 to max_data-1  do
      begin
            sum.re := 0.0 ;
            sum.im := 0.0 ;


            for  j := 0 to max_data-1  do
            begin
              sum.re := sum.re + u^[i]^[j].re ;
              sum.im := sum.im + u^[i]^[j].im ;
            end {loop} ;

            sum.re := sum.re / max_data ;
            sum.im := sum.im / max_data ;
```

```
          for  j := 0 to max_data-1  do
          begin
            u^[i]^[j].re :=  u^[i]^[j].re - sum.re ;
            u^[i]^[j].im :=  u^[i]^[j].im - sum.im ;
          end {loop} ;

     end {loop} ;

end {reduce_dc_from_matrix} ;


procedure ask_for_smoothing ( var u : matrix ) ;
var
   inkey : char ;
begin
     repeat
          write (' Want to smoothing signal ? (y/n) ') ;
          inkey := readkey ; cr;cr;
          if inkey in ['y','Y',esc] then
            begin
               writeln (' Smoothing signal . ') ;cr;
               start;
               smoothing_matrix ( u ) ;
               stop ;
               writeln(' Time used : ',base_60(time_used))
;
            end
          else
            if inkey in ['n','N',ret] then
               writeln(' Signal was not smoothing .') ;
           {end if}
         {end if} ;
     until inkey in ['n','N','y','Y',ret,esc] ;
     cr;udl;cr;
end {ask_for_smoothing} ;


procedure  ask_for_reduce_DC_from ( var u : matrix) ;
var
   inkey : char ;
begin
     repeat
          write (' Want to reduce DC from signal ? (y/n) ')
;
          inkey := readkey ; cr;cr;
          if inkey in ['y','Y',esc] then
            begin
               writeln (' Reduce DC from signal . ') ;cr;
               start;
               reduce_dc_from_matrix (u) ;
               stop ;
               writeln(' Time used : ',base_60(time_used))
;
            end
```

```
          else
            if inkey in ['n','N',ret] then
               writeln(' DC does not reduce from signal .')
;
           {end if}
         {end if} ;
     until inkey in ['n','N','y','Y',ret,esc] ;
     cr;udl;cr;
end {ask_for_reduce_DC_from} ;


procedure  filtering_matrix ( var x : matrix ) ;
var
     item  :  byte ;
     inkey :  char    ;
     attr_save : byte ;
const
     esc    = #27 ;
     return = #13 ;
begin

     open_window ( filter_wind ) ;
     repeat
          clrscr;
          highvideo ;

          select_menu ( filter_menu ,item ,inkey ,{at} 1,1
) ;

          HighVideo;

          attr_save := TextAttr ;
          TextAttr  := $1E ;
          clrscr ;
          gotoxy (5,2) ;
          write (' DOING ') ;
        if  inkey = return  then
            case  item  of
                    1  :  smoothing_matrix ( x ) ;
                    2  :  reduce_dc_from_matrix ( x ) ;
            end {case}
          {end if} ;
          TextAttr := attr_save ;
          LowVideo;

     until  (inkey = esc) or ((inkey = return) and (item =
3))
           or (last_menu_scan_key=RightArrow)
           or (last_menu_scan_key=LeftArrow) ;

     close_window ;

end ;
```

```
begin  { init section }
     new_menu ( filter_menu ) ;
     menu_normal_color    ( filter_menu , Yellow ,Blue ) ;
     menu_inverse_color   ( filter_menu , White , Magenta )
;
     menu_highlight_color ( filter_menu , LightCyan ,Blue )
;

     add_menu ( filter_menu ,'S', '  Smooth      ') ;
     add_menu ( filter_menu ,'R', '  Reduce DC   ') ;
     add_menu ( filter_menu ,'E', '  Exit        ') ;

     define_window  ( filter_wind , 13,10 , 27,14 ) ;
     window_color   ( filter_wind , LightGray , Blue ) ;
     broader_color  ( filter_wind , White , Cyan ) ;
     window_heading ( filter_wind , ' SIGNAL FILTER ') ;
     head_color     ( filter_wind , LightCyan , Cyan ) ;
end..
```

ภาคผนวก ฉ

ยูนิต askfname

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการถามชื่อไฟล์จากผู้ใช้โดยที่ผู้ใช้อาจเลือกใช้วิธีเลือกจากรายการที่แสดง

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจเพียงคำสั่งเดียวคือโพรซิเยอร์ ask_for_file_name มีพารามิเตอร์ใช้ส่งชื่อไฟล์ที่ผู้ใช้เลือกกลับ โดยที่ในขั้นแรกเมื่อโพรซิเยอร์นี้ถูกเรียก จะถามชื่อไฟล์จากผู้ใช้ ถ้าหากว่าผู้ใช้กดคานเว้นวรรคแล้วกดปุ่ม Enter โปรแกรมจะแสดงชื่อไฟล์ออกมาให้เลือก ดังในภาพ ก.4

โปรแกรมได้แสดงไว้ในหน้าถัดไป

```
unit askFname ;

interface

uses  dos,crt,cell,windsys,mouse ;

procedure ask_for_file_name ( var file_spec : string ; x,y :
byte ) ;

implementation


const
     menu_col   =  4 ;
     menu_row   =  9 ;
     max_col    =  4 ;
     max_row    = 40 ;
     max_count  = 160;
     null       = '' ;
     no_path    = '';
     blank = ' ' ;
     colon      = ':';
     name_size  = 16 ;
     slash      = '\';
     NoError    = 0 ;
     NUL        = #00 ;


     dir_fore_color : byte =  Blue ;
     dir_back_color : byte =  LightGray ;

     head_fore_color : byte =  White ;
     head_back_color : byte =  LightGray ;


     ask_fore_color : byte =  White ;
     ask_back_color : byte =  Blue ;

     pointer_fore_color : byte = Yellow;
     pointer_back_color : byte = Magenta;


type
      name_string  = string[name_size] ;
      file_info    = record
                          name  :  name_string ;
                          attr  :  byte ;
                     end ;
      double_array = array [1..max_row,1..max_col] of
file_info ;
      single_array = array [1..max_count] of file_info ;
```

```
      disk_info    = record
                     case integer  of
                           1:( sing : single_array ) ;
                           2:( doub : double_array ) ;
                     end ;

var
      catalog    :  ^disk_info ;
      disk_size :  word  ;
      over_area :  boolean  ;

procedure inverse ;
begin
     TextAttr  := pointer_fore_color + pointer_back_color
*16 ;
end ;

procedure normal ;
begin
     TextAttr  := dir_fore_color + dir_back_color * 16 ;
end ;

function space( n : byte ) : string ;
var x : string ;
    a : byte absolute x ;
begin
     FillChar ( x , n+1 , #32 ) ;
     a := n ;
     space := x ;
end ;

procedure  clean_catalog ;
var i : integer ;
begin
     for i := 1 to max_count do
     begin
          catalog^.sing[i].name := null ;
          catalog^.sing[i].attr := 0    ;
     end ;
end;


procedure  create_catalog ;
var  mem_need : word ;
     i        : byte ;
begin
     mem_need :=  SizeOf (disk_info) ;
     if  MaxAvail > mem_need  then
         new (catalog)
     else
         out_of_memory
     {end if} ;
     clean_catalog ;
end ;
```

```
procedure  cancel_catalog ;
begin
     dispose (catalog) ;
end ;

procedure read_directory ( path , wildcards : string ) ;
var  file_name    : name_string ;
     file_attr    : byte ;
     information : SearchRec ;
     count,i,j    : integer ;
begin

     count := 1 ;
     FindFirst( path + wildcards , Directory + archive ,
information ) ;

     while (DosError = 0) and (count <= max_count)do
     begin
          catalog^.sing[count].name :=  information.name ;
          catalog^.sing[count].attr :=  information.attr ;
          inc( count ) ;
          FindNext ( information ) ;
     end {while} ;

     if count > max_count then
        over_area := true
     else
       over_area := false
     {end if} ;
     disk_size := count-1 ;
end ;

procedure  swap_info ( var  a,b : file_info ) ;
var
   temp  : file_info ;
   count : word ;
begin
     temp := a ;
     a := b ;
     b := temp ;
end ;


procedure  sort_catalog ;
var
   count : word ;
   swap_occur : boolean ;
begin
     if  disk_size = 0  then  exit ;
```

```
     { Sort on name }
     with  catalog^  do
     repeat
           swap_occur := false ;
           for count := 1 to disk_size-1 do
           begin
           if  sing[count].name > sing[count+1].name  then
                begin
                     swap_info ( sing[count] , sing[count+1]
) ;
                     swap_occur := true ;
                end {if} ;
           end {loop} ;
     until not swap_occur ;

     { Sort on attribute }
     with  catalog^  do
     repeat
           swap_occur := false ;
           for count := 1 to disk_size-1 do
           begin
                if  sing[count].attr < sing[count+1].attr
then
                begin
                     swap_info ( sing[count] , sing[count+1]
) ;
                     swap_occur := true ;
                end {if} ;
           end {loop} ;
     until not swap_occur ;
end ;


function name_of ( page , i,j : byte ) : string ;
var  name : name_string ;
     attr : byte ;
begin
     name := catalog^.doub [i+page-1][j].name ;
     attr := catalog^.doub [i+page-1][j].attr ;
     if  attr = Directory  then  name :=  name + slash ;
     name := blank + name ;
     name_of := name + space ( name_size-length(name)-1 ) ;
end ;


procedure show_page ( page ,{at} x,y : byte ) ;
var
   i,j  :  word ;
   name :  name_string ;
begin
```

```
      for i := 1 to menu_row do
      for j := 1 to menu_col do
      begin
           name := catalog^.doub[i+page-1,j].name ;
           gotoxy ( x + (j-1)*name_size , y + i-1 ) ;

           if  name = null  then
               write( space(name_size-1))
           else
               write( name_of ( page , i,j) )
           {end if};
      end ;
end ;


procedure  check_range ( var page ,i,j : integer ) ;
begin
            if j<1 then
               if (i=1) and (page=1) then
                  j := 1
               else
                  begin dec(i) ; j := menu_col ; end
               {end if}
            else if j > menu_col then
               begin inc(i) ; j := 1 ; end
            {end if} ;

            if i<1 then
               begin  dec(page) ; i := 1  end
            else if i > menu_row then
               begin  inc(page) ; i := menu_row  end
            {end if};

            if page<1 then
               page := 1
            else if (page > ((disk_size-1) div menu_col)-
menu_row+2)  then
                     if  (disk_size > menu_row*menu_col )
then
                        page := ((disk_size-1) div menu_col)-
menu_row+2
                     else
                        page := 1
                     {end if} ;
            {end if} ;

            while catalog^.doub [i+page-1][j].name = null do
dec(i) ;

end {check_range} ;
```

```
const  dir_wind  :  window_type = nil ;



procedure set_dir_wind_head ;
var path : string ;
begin
     GetDir (0,path) ;
     window_heading ( dir_wind , blank+path+blank ) ;

     if over_area then
          bottom_line ( dir_wind ,' Too many files. ')
     else
          bottom_line ( dir_wind ,'')
     {end if} ;

     broader_and_head ( dir_wind) ;
end ;



procedure  extract_filespec ( var file_spec ,{to} path
,{and} name : string ) ;
var
   filespec_is_path_name : boolean ;
   name_pos              : integer ;
begin
     filespec_is_path_name :=  exist_path (file_spec) ;

     if  filespec_is_path_name  then
       begin
          path := file_spec ;
          name := '*.*' ;
       end
     else
       begin { extract path_name and file_name }
          name := file_spec ;
          while  (pos(colon,name)<>0) or
(pos(slash,name)<>0)  do
              delete(name,1,1)
          {end while};
          name_pos := pos( name , file_spec ) ;
          path := file_spec ;
       if  name_pos<>0  then  delete
(path,name_pos,length(name));
       if  copy ( path ,length(path),1 ) = slash  then
            delete (path,length(path),1 ) ;
       end
     {end if} ;
end {extract filespec} ;
```

```
procedure select_file_of ( var  file_spec : string ; x,y :
byte ) ;
const
     DefaultDrive  = 0 ;

var
     page          ,
     old_page      ,
     name_pos      ,
     mouse_pos_x   ,
     mouse_pos_y   ,
     i,j,io,jo     : integer ;

     scan          ,
     inkey         : char ;
     attr          ,
     attr_save     : byte ;
     drive         ,
     path          ,
     name          ,

     path_save1    ,
     path_save2    ,
     wildcards     : string ;
     again         : boolean ;

     keyboard_active    ,
     mouse_active       ,
     mid_button         ,
     left_button        ,
     no_file_select     ,
     one_file_select    ,
     path_select        :  boolean ;


          procedure point_item ;
          var  name : name_string ;
               attr : byte ;
          begin
               if page <> old_page then  show_page ( page
,{at} x,y ) ;

               gotoxy( x + (jo-1)*name_size , io+y-1 );
               write( name_of ( page , io,jo) );
               inverse;
               gotoxy( x + (j-1)*name_size , i+y-1 );
               write( name_of ( page , i,j) );
               normal ;
               gotoxy( x + (j-1)*name_size , i+y-1 );
```

```
                io := i ; jo := j ; old_page := page ;

            end {point_at} ;


            procedure  save_old_path ;
            begin
                path_save1 :=  null ;
                path_save2 :=  null ;

                GetDir (DefaultDrive, path_save1) ;
                if  pos(colon,path)=2 { path to other drive }
then
                begin
                     drive := copy (path,1,2) ;
                     ChDir (drive) ;
                if IoResult = NoError then GetDir
(DefaultDrive,

path_save2) ;
                end {if} ;
            end  {save_old_path} ;


            procedure  restore_old_path ;
            begin
                if  path_save2 <> null  then ChDir
(path_save2) ;
                if  path_save1 <> null  then ChDir
(path_save1) ;
            end  {restore_old_path} ;


            procedure  combine_from_path_and_name_to_file_spec
;
            begin
                if  length(path) = 3  then  delete (path,3,1)
;

                if  one_file_select  then
                   if path = null then
                     file_spec :=  name
                   else
                     file_spec := path + slash + name
                   {end if}
                {end if} ;
            end ;

            procedure  change_path_to ( name : string ) ;
            begin
                clean_catalog;
                clrscr;
```

```
                if  exist_path (name) then  Chdir (name) ;

                read_directory ( no_path , wildcards ) ;
                sort_catalog ;
                set_dir_wind_head ;
                i  := 1 ;  j  := 1 ;  page     := 1 ;
                io := 1 ;  jo := 1 ;  old_page := 2 ;
           end {change_path} ;


       procedure  convert_to_line_column ( var mouse_pos_x ,
mouse_pos_y : integer ) ;
           begin
                 mouse_pos_x := (mouse_pos_x div 8)  ;
                 mouse_pos_x :=  mouse_pos_x - Lo(WindMin) +
1 ;
                 mouse_pos_y := (mouse_pos_y div 8)  ;
                 mouse_pos_y :=  mouse_pos_y - Hi(WindMin) +
1 ;
           end ;

           function  col_mouse_on : integer ;
           begin
                col_mouse_on := ((mouse_pos_x-1) div
name_size) + 1  ;
           end ;


           function  pos_x_and_y_in_range :  boolean ;
           begin
                pos_x_and_y_in_range :=  (mouse_pos_x > 0)
                                         and (mouse_pos_x <
menu_col*name_size+1)
                                         and (mouse_pos_y > 0)
                                         and (mouse_pos_y <
menu_row+1) ;
           end ;

           function  mouse_on_pointer : boolean ;
           begin
                mouse_on_pointer  :=  (col_mouse_on = j) and
                                      (mouse_pos_y  = i) ;
           end ;

begin
     define_window ( dir_wind , x,y ,
x+menu_col*name_size+1,y+menu_row+1 );
     window_color  ( dir_wind , dir_fore_color ,
dir_back_color ) ;
     broader_color ( dir_wind , dir_fore_color ,
dir_back_color ) ;
```

```
     head_color     ( dir_wind , head_fore_color ,
head_back_color ) ;

     x:=1 ; y:=1 ;

     Extract_FileSpec ( {from} file_spec ,{to} path ,{and}
name ) ;
     { Now , we get path and filename suparately }


     save_old_path ;

     wildcards := name ;

     Chdir( path )  ; { change path to input path }

     create_catalog ; { create area for store directory data
}
     read_directory ( no_path , wildcards ) ;

     if  disk_size = 0  { no any file on this directory }
then
     begin
          file_spec := null ;
          exit ;
     end {if} ;

     sort_catalog ;

     i  := 1 ;  j  := 1 ;  page     := 1 ;
     io := 1 ;  jo := 1 ;  old_page := 2 ;

     open_window ( dir_wind ) ; clrscr ;

     set_dir_wind_head;

     point_item  ;

     repeat

{ifdef use_mouse}

        clear_button_press ;
           wait_mouse_or_key ;

           if  pressing_mouse_button (left)  then
           begin
                get_press_pos_of_button (left
,mouse_pos_x,mouse_pos_y ) ;
                convert_to_line_column  ( mouse_pos_x ,
mouse_pos_y ) ;
```

```
                if mouse_on_pointer then
                  begin
                       name :=  catalog^.doub [i+page-
1][j].name ;
                       attr :=  catalog^.doub [i+page-
1][j].attr ;

                       path_select     :=  (attr =
Directory) ;
                       one_file_select :=  (attr = Archive)
;
                       no_file_select  :=  false ;

                       if  path_select  then
                       begin
                            change_path_to (name) ;
                            point_item ;
                       end {if} ;
                  end
                else
                  begin
                     show_icon ;
                     repeat
                          mouse_active  :=
pressing_mouse_button (left) ;
                          get_press_pos_of_button (left ,
mouse_pos_x,mouse_pos_y );
                          convert_to_line_column (
mouse_pos_x , mouse_pos_y );

                          if pos_x_and_y_in_range and not
mouse_on_pointer
                          then
                          begin
                               j := col_mouse_on ;
                               i := mouse_pos_y  ;
                          end {if} ;

                          if  mouse_pos_y = 0  then  dec
(page) ;
                          if  mouse_pos_y = (menu_row+1)
then inc (page) ;

                          check_range (page ,i,j ) ;
                          if (page<>old_page) or (i<>io) or
(j<>jo)  then
                          begin
                               hide_icon  ;
                               point_item ;
                               show_icon ;
                          end {if} ;

                     until not mouse_active ;
```

```
                    hide_icon ;

                    one_file_select := false ;
                    no_file_select  := false ;

                  end {if}
                {end if} ;
        end {if} ;


            if pressing_mouse_button (middle) then
            begin
                 path_select     :=  false ;
                 one_file_select :=  false ;
                 no_file_select  :=  true  ;
            end ;
{endif}

            if  KeyPressed  then
            begin
                 scan_key_to ( scan , inkey ) ;

                 case  scan  of
                     UpArrow  :  dec(i) ;
                    DownArrow :  inc(i) ;
                    LeftArrow :  dec(j) ;
                   RightArrow :  inc(j) ;
                      PgUp    :  dec(page,(menu_row-1) ) ;
                      PgDn    :  inc(page,(menu_row-1) ) ;
                      Home    :  begin  page := 1 ; i := 1 ;
j := 1 ;  end ;
                      EndKey  :  begin
                                    page := (disk_size-1) div
menu_col + 1 ;

                                    i    := menu_row ;
                                    j    := (disk_size-1) mod
menu_col + 1 ;
                                 end ;
                 end {case} ;

                 check_range ( page ,i,j ) ;

                 name :=  catalog^.doub [i+page-1][j].name ;
                 attr :=  catalog^.doub [i+page-1][j].attr ;

                 path_select     :=  (inkey = return) and
(attr = Directory) ;
                 one_file_select :=  (inkey = return) and
(attr = Archive)   ;

                 no_file_select  :=  (inkey = Esc) ;
```

```
                if  path_select  then  change_path_to (name)
;

                point_item  ;

            end {if} ;

      until  no_file_select  or  one_file_select  ;


      GetDir ( 0, path ) ; { get path selected }

      restore_old_path ;

      { combine  path & name --> file_spec }

      if  no_file_select  then  begin path := null ; name :=
null ; end ;

      if  one_file_select then
combine_from_path_and_name_to_file_spec ;

      { Ok, now we get file spec of selected file }

      { if no any file selected file_spec is null }

      close_window ;
      cancel_catalog ;
      cancel_window( dir_wind ) ;

end {select_file_of} ;


procedure ask_for_file_name ( var file_spec : string ; x,y :
byte ) ;
const
      ask_wind  :  window_type = nil ;
      colon     =  ':' ;
      slash     =  '\' ;
var
     information : SearchRec ;
     name_save ,
     path_save ,
     name      ,
     path      : string ;
     attr_save ,
     name_pos  : byte ;
     inkey     : char ;

     name_is_wildcard  : boolean ;
     name_is_path_name : boolean ;
```

```
begin
     define_window  ( ask_wind , x,y , x+28,y+2 ) ;
     window_heading ( ask_wind ,' Enter file name ');
     window_color   ( ask_wind , ask_fore_color ,
ask_back_color ) ;
     broader_color  ( ask_wind , ask_fore_color ,
ask_back_color ) ;
     head_color     ( ask_wind , head_fore_color ,
head_back_color ) ;
     open_window ( ask_wind ) ; clrscr ;

     gotoxy(1,1);
     name := file_spec ;  default_read(name) ;

     while pos(blank,name)<>0 do
delete(name,pos(blank,name),1) ;
     {eliminate blank }

     name_is_wildcard  := (pos('*',name)<>0) or
(pos('?',name)<>0) ;

     name_is_path_name :=  exist_path (name) ;


     if  (name = null) or (name_is_wildcard) or
(name_is_path_name) then
       begin
          select_file_of ( name , 8 ,7 ) { from directory }
;

          if  name = null { no file selected } then
             file_spec :=  null
          else
             file_spec :=  name
          {end if} ;
       end
     else
       file_spec  := name ;
     {end if} ;


     close_window ;
     cancel_window( ask_wind );
end;

begin



end..
```

ภาคผนวก ข

ยูนิต cell

ในยูนิตนี้จะเป็นคำสั่งที่สร้างขึ้นเพื่อใช้ในงานทั่วๆไปซึ่งเป็นงานย่อยเล็กๆน้อยๆ และไม่สะดวกที่จะจัดเป็นยูนิตต่างหาก

ตัวโปรแกรมได้แสดงไว้โดยละเอียดต่อจากบทความนี้แล้ว

คำสั่งที่สร้างในยูนิต

ในยูนิตนี้มีคำสั่งที่น่าสนใจอยู่มากมายดังต่อไปนี้

1. โพรซิเยอร์ cr ไม่มีพารามิเตอร์ ใช้สั่งให้ขึ้นบรรทัดพิมพ์ใหม่บนจอภาพ

2. โพรซิเยอร์ wait ไม่มีพารามิเตอร์ ใช้สั่งให้หยุดการทำงานของโปรแกรมและรอจนกว่าจะมีการกดปุ่มใดๆบนแป้นพิมพ์จึงจะทำงานต่อไป

3. โพรซิเยอร์ call ไม่มีพารามิเตอร์ ใช้สั่งให้หยุดการทำงานของโปรแกรมและส่งเสียงเตือน พร้อมกับรอจนกว่าจะมีการกดปุ่มใดๆบนแป้นพิมพ์จึงจะหยุดส่งเสียงและทำงานต่อไป

4. โพรซิเยอร์ click ไม่มีพารามิเตอร์ ใช้สั่งให้ส่งเสียง คลิก เบาๆหนึ่งครั้ง

5. โพรซิเยอร์ out_of_memory ไม่มีพารามิเตอร์ จะแสดงข้อความเตือนว่าหน่วยความจำไม่พอเพียงแก่การทำงาน หยุดการทำงานของโปรแกรมและส่งเสียงเตือน พร้อม

กับรอจนกว่าจะมีการกดปุ่มใดๆบนแป้นพิมพ์จึงจะหยุดส่งเสียงและยุติการทำงานของโปรแกรม

6. โพรซีเยอร์ waiter ไม่มีพารามิเตอร์ จะแสดงข้อความเตือนว่า 'Press any key for continue' และจะหยุดการทำงานของโปรแกรมรอจนกว่าจะมีการกดปุ่มใดๆ บนแป้นพิมพ์จึงจะทำงานต่อไป

7. โพรซีเยอร์ udl ไม่มีพารามิเตอร์ จะทำการขีดเส้นจากขอบซ้ายไปยังขอบขวาของจอภาพ

8. โพรซีเยอร์ writeat มีพารามิเตอร์เป็นตำแหน่งบนจอภาพ และข้อความ จะแสดงข้อความออกบนจอภาพ ณ ตำแหน่งที่กำหนด

9. ฟังชัน display_page จะส่งค่าหมายเลขจอภาพผ่านทางชื่อฟังชัน

10. โพรซีเยอร์ OutChar เป็นโพรซีเยอร์ที่ใช้ส่งตัวอักษรออกทางจอภาพโดยตรง

11. โพรซีเยอร์ OutString เป็นโพรซีเยอร์ที่ใช้ส่งข้อความออกทางจอภาพโดยตรง

12. โพรซีเยอร์ hide_cursor ใช้สั่งให้เคอร์เซอร์หายไป

13. โพรซีเยอร์ show_cursor ใช้สั่งให้เคอร์เซอร์กลับปรากฏขึ้น

14. โพรซีเยอร์ spooling_file มีสองพารามิเตอร์คือ file_name และ device ซึ่งเป็นชนิดข้อความทั้งคู่ โพรซีเยอร์นี้จะส่งข้อความที่อยู่ในไฟล์ตามที่กำหนดในพารามิเตอร์ file_name ออกไปยังอุปกรณ์ที่ติดตั้งอยู่ที่กำหนดโดยพารามิเตอร์ device โดยที่การส่งข้อความนี้จะเรียกใช้โปรแกรม print.exe ซึ่งจะช่วยให้สามารถทำงานอื่นๆต่อไปได้เลยโดยไม่ต้องรอให้การส่งข้อมูลเรียบร้อยก่อน

15. โพรซีเยอร์ sc key_to มีพารามิเตอร์ส่งกลับสองตัวคือ scan_code และ ascii_code ซึ่งเป็นชนิดตัวอักษรทั้งคู่ โพรซีเยอร์นี้หยุดรอการกดปุ่มบนแป้นพิมพ์ และจะส่งค่า scan code และ ASCII code ของปุ่มที่ถูกกดผ่านกลับมาทางพารามิเตอร์

16. ฟังชัน key_flag จะส่งค่าแฟลกของแป้นพิมพ์ผ่านทางชื่อฟังชัน

17. ฟังชัน flag_of_key มีพารามิเตอร์เป็นแฟลกของแป้นพิมพ์ที่ต้องการทดสอบ จะส่งผลผ่านทางชื่อฟังชัน

18. ฟังชัน upper_case มีพารามิเตอร์เป็นตัวอักษร จะเปลี่ยนตัวอักษรในภาษาอังกฤษเป็นตัวใหญ่ แล้วส่งผลผ่านทางชื่อฟังชัน

19. โพรซีเยอร์ print_screen จะทำสำเนาของข้อความที่ปรากฏบนจอภาพออกทางเครื่องพิมพ์

20. โพรซีเยอร์ broader มีพารามิเตอร์สี่ตัวคือ x1,y1 ,x2,y2 ซึ่งระบุตำแหน่งบนจอภาพสองตำแหน่ง โพซีเยอร์นี้จะตีกรอบสี่เหลี่ยมบนจอภาพ โดยมีมุมอยู่ที่ตำแหน่งที่กำหนด

21. โพรซีเยอร์ screen_broader ไม่มีพารามิเตอร์ จะตีกรอบสี่เหลี่ยมรอบจอภาพ

22. ฟังชัน exist_file มีพารามิเตอร์เป็นชื่อไฟล์ จะทดสอบว่ามีไฟล์นี้อยู่หรือไม่ แล้วส่งผลผ่านทางชื่อฟังชัน

23. ฟังชัน exist_path มีพารามิเตอร์เป็นชื่อไดเรคเทอรี่ จะทดสอบว่ามีไดเรคเทอรี่นี้อยู่หรือไม่ แล้วส่งผลผ่านทางชื่อฟังชัน

24. โพรซีเยอร์ terminate_to_os_shell ไม่มีพารามิเตอร์ จะออกจากโปร

แกรมไปทำงานที่ MS-DOS และสามารถกลับเข้ามาทำงานที่เดิมได้โดยใช้คำสั่ง EXIT ที่ MS-DOS

25. โพรซีเยอร์ default_read มีพารามิเตอร์เป็นข้อความ จะแสดงข้อความที่ส่งผ่านพารามิเตอร์เข้ามาก่อน ถ้าหากผู้ใช้กดปุ่ม Enter ก็จะส่งข้อความนั้นกลับผ่านพารามิเตอร์ แต่ถ้ากดปุ่มอื่นๆก็จะอ่านข้อความจากทางแป้นพิมพ์แล้วส่งกลับผ่านพารามิเตอร์

26. โพรซีเยอร์ get_display_mode มีพารามิเตอร์ส่งกลับสามตัว ซึ่งจะใช้ส่งกลับ ชนิด หมายเลขหน้า และความกว้างเป็นคอลัมน์ของจอภาพในขณะนั้น

27. โพรซีเยอร์ get_screen_ptr มีพารามิเตอร์ส่งกลับหนึ่งตัว ซึ่งจะใช้ส่งกลับตำแหน่งของหน่วยความจำที่เป็นของจอภาพ

28. โพรซีเยอร์ save_screen_to มีพารามิเตอร์ส่งกลับหนึ่งตัว โพรซีเยอร์นี้จะบันทึกจอภาพลงในหน่วยความจำ และส่งกลับตำแหน่งของหน่วยความจำที่ใช้ผ่านทางพารามิเตอร์

29. โพรซีเยอร์ restore_screen_from มีพารามิเตอร์ส่งหนึ่งตัว โพรซีเยอร์นี้จะนำจอภาพที่บันทึกไว้ในหน่วยความจำตามตำแหน่งของหน่วยความจำที่ระบุผ่านทางพารามิเตอร์ ไปแสดงออกยังจอภาพ

```
unit cell ;

interface

uses  dos,crt,mouse ;

const   InsertOn      = $80 ;       { These consanst is used
with }
        CapsLockOn    = $40 ;       { function 'key_flag'
}
        NumLockOn     = $20 ;
        ScrollLockOn  = $10 ;
        AltDown       = $08 ;
        CtrlDown      = $04 ;
        LshiftDown    = $02 ;
        RshiftDown    = $01 ;

        NoError       = 0 ;

const   Home          = #$47 ;        { These is scan code of
numeric key }
        UpArrow       = #$48 ;
        PgUp          = #$49 ;
        MinusSign     = #$4A ;
        LeftArrow     = #$4B ;
        FiveKey       = #$4C ;
        Rightarrow    = #$4D ;
        PlusSign      = #$4E ;
        EndKey        = #$4F ;
        DownArrow     = #$50 ;
        PgDn          = #$51 ;
        Ins           = #$52 ;
        Del           = #$53 ;

const
        null    =  #00 ;              { Often used key code }
        return  =  #13 ;
        esc     =  #27 ;
        space   =  #32 ;
        back_space = #08 ;

        null_string = '' ;

type intptr         =  ^integer ;
     screen_type    =  ^integer ;


procedure   cr ;
procedure   wait ;
procedure   call ;
procedure   click ;

procedure   out_of_memory ;
```

```
procedure   waiter ;

procedure   udl ;

procedure   writeat( x,y : byte ; a : string ) ;

function    display_page : byte ;

procedure   OutChar ( c : char ) ;

procedure   OutString ( a : string ) ;

procedure   hide_cursor ;

procedure   show_cursor ;


procedure   spooling_file ( file_name {to}, device : string
) ;

procedure   scan_key_to ( var scan_code , ascii_code  :
char ) ;

function    key_flag : byte ;

function    flag_of_key ( x : byte ) : boolean ;

function    upper_case ( x : char ) : char ;

procedure   print_screen ;

procedure   broader ( x1,y1, x2,y2 : integer ) ;

procedure   screen_broader ;

function    exist_file ( file_name : string ) : boolean ;

function    exist_path ( path_name : string ) : boolean ;


procedure   terminate_to_os_shell ;

procedure   default_read ( var a : string ) ;

procedure   get_display_mode ( var  mode , page , col : byte
) ;

procedure   set_screen_ptr   ( var scrptr : intptr ) ;

procedure   save_screen_to ( var screen_save : screen_type )
;
```

```
procedure   restore_screen_from ( var screen_save :
screen_type )  ;


var  last_scan_key : char ;
     last_inkey    : char ;

implementation


const max_col = 80 ;
      max_line = 25 ;

      video_service = $10 ;

const
      cursor_save  :  record
                           starting,ending : byte ;
                           inused          : boolean ;
                      end
                   =  (starting:0;ending:0;
                       inused:false);


procedure   cr ; begin  write(#13#10)  end;
procedure wait ; var x,y:char; begin  scan_key_to (x,y) ;
end ;
procedure call ;
var i : real ;
begin
     repeat
           i := i + 0.001 ;
           if i > pi-0.1 then i := 0 ;
           sound( round(1000*(2+sin(i)) ) )  ;
           delay(1)
     until keypressed ;
     nosound ;
     wait ;
end ;

procedure   click ;
begin
     sound(5000) ;
     delay(5) ;
     nosound ;
end ;

procedure  out_of_memory ;
begin
     writeln( ' ****  OUT OF MEMORY  **** ' ) ; cr;
     writeln( ' !!! program terminate !!! ' ) ; cr;cr;
     call ;
     Halt ;
end ;
```

```
procedure waiter ;
begin
     writeln(' Press any key for continue .') ;
     wait ;
end ;

procedure udl ;
var  i : integer ;
begin
     for i := 0 to Lo(windmax)-Lo(windmin) do write('=');
end;


procedure  scan_key_to ( var scan_code , ascii_code  :  char
) ;
var  reg    : registers ;
begin
     wait_mouse_or_key ;

     if  KeyPressed  then
       begin
        { get key } reg.ah  :=  0 ; intr($16,reg) ;
          scan_code   :=  char(reg.ah) ;
          ascii_code  :=  char(reg.al) ;
       end
     else
       begin
          if pressing_mouse_button (middle) then
          begin
               scan_code  := #01 ;
               ascii_code := #27 ;
          end {if} ;

          if pressing_mouse_button (left) then
          begin
               scan_code  := #$1C ;
               ascii_code := #13 ;
          end {if} ;

          if pressing_mouse_button (right) then
          begin
               scan_code  := #$39 ;
               ascii_code := #32 ;
          end {if} ;

     end {if} ;
```

```
     last_scan_key := scan_code  ;
     last_inkey    := ascii_code  ;
end ;


function key_flag : byte ;
var
    regs : registers ;
begin
     regs.AH := 02 ;
     intr ( $16 , regs ) ;
     key_flag := regs.AL ;
end ;


function     flag_of_key ( x : byte ) : boolean ;
begin
     if  (key_flag and x) <> 0 then
         flag_of_key := true
     else
         flag_of_key := false
     {end if} ;
end ;



function upper_case ( x : char ) : char ;
begin
     case  x  of  'a'..'z'  :  dec(x,32) ;  end {case} ;
     upper_case := x ;
end ;



procedure print_screen ;
var  reg : registers ;
begin
     intr(5,reg) ;
end ;


procedure   writeat( x,y : byte ; a : string ) ;
begin
     gotoxy(x,y) ;
     write(a) ;
end ;
```

```
function display_page : byte ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $0F ;
          intr ( video_service , regs ) ;
          display_page := BH ;
     end {with} ;
end ;


procedure  OutChar ( c : char ) ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $09 ;
          AL := byte(c) ;
          BH := display_page ;
          BL := Crt.TextAttr ;
          CX := 1 ;
          intr ( video_service , regs ) ;
     end {with} ;
end ;

procedure  move_cursor ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $03 ;
          BH := display_page ;
          intr ( video_service , regs ) ; { Get cursor
position }
          AH := $02 ;
          inc (DL) ;  {increment column}
          BH := display_page ;
          intr ( video_service , regs ) ; { Set cursor
position }
     end {with} ;
end ;



procedure  OutString ( a : string ) ;
var
   len  :  byte   absolute  a  ;
   s    :  array [0..255] of char absolute  a ;
   i    :  integer  ;
begin
```

```
      if  len > 1  then
        begin
           dec (len) ;
           write (a) ;
           OutChar ( s[len+1] ) ;
        end
      else
        OutChar (s[i]) ;


end ;

procedure  hide_cursor ;
var   regs : Registers ;
begin
      if  not cursor_save.inused  then
      with  regs  do
      begin
           AH := $03 ;
           BH := display_page ;
           intr ( video_service , regs ) ; {Read cursor type}
           cursor_save.starting := CH ;
           cursor_save.ending   := CL ;

           AH := $01 ;
           BH := display_page ;
           CH := 13 ;
           CL := 14 ;
           intr ( video_service , regs ) ; {Set cursor hide}
      end {if} ;
end ;

procedure  show_cursor ;
var   regs : Registers ;
begin
      if  cursor_save.inused  then
      with  regs  do
      begin
           AH := $01 ;
           BH := display_page ;
           CH := cursor_save.starting ;
           CL := cursor_save.ending   ;
           intr ( video_service , regs ) ; {Restore cursor}
      end {if} ;
end ;



procedure   spooling_file ( file_name {to}, device : string
) ;
var      spool       : string ;
const    screen_save : screen_type = nil ;
```

```
begin
     save_screen_to (screen_save)  ;
     if  exist_file ( file_name )  then
         begin
              spool := '/C c:\msdos\print /d:lpt2 ' +
file_name ;
              exec   ( 'C:\COMMAND.COM' , spool ) ;
         end
     else
         begin
              writeln ( ' file '+file_name+
                        ' does not exist , cannot do
background print .');
              wait;
         end
     {end if} ;
     restore_screen_from (screen_save)  ;

end {print_file};



procedure broader ( x1,y1, x2,y2 : integer ) ;
const
     ulc = #$C9 ;  us  = #$CD ;  urc = #$BB ;
     ls  = #$BA ;                rs  = #$BA ;
     llc = #$C8 ;  ds  = #$CD ;  lrc = #$BC ;
var
     i  :  integer ;
begin
     gotoxy(x1,y1) ; write (ulc) ;
     for i := x1+1 to x2-1 do write(us) ;
     gotoxy(x2,y1) ; write (urc) ;
     for i := y1+1 to y2-1 do
     begin
          gotoxy(x1,i) ; write(ls) ;
          gotoxy(x2,i) ; write(rs) ;
     end;
     gotoxy(x1,y2) ; write (llc) ;
     for i := x1+1 to x2-1 do write(ds) ;
     gotoxy(x2,y2) ; write (lrc) ;
end;


procedure screen_broader ;
type
    coor = record  x,y : byte  end ;
var
    windmin : coor  absolute  Crt.Windmin ;
    windmax : coor  absolute  Crt.Windmax ;
```

```
begin
     broader ( windmin.x+1,windmin.y+1,windmax.x+1,windmax.y
) ;
end ;


function exist_file ( file_name : string ) : boolean ;
var
    information : SearchRec ;
    name        : string ;
const
     star = '*' ;
     ques = '?' ;
begin
     name := file_name ;
     FindFirst ( name , Archive , information ) ;
     if (DosError = NoError) and (pos(star,name)=0) and
(pos(ques,name)=0)
     then
         exist_file := true
     else
         exist_file := false
     {end if} ;
end ;


function exist_path ( path_name : string ) : boolean ;
var
    information : SearchRec ;
    name        : string ;
    drive       : string ;
    path_save1  : string ;
    path_save2  : string ;
const
     null  = ''  ;
     colon = ':' ;
     star  = '*' ;
     ques  = '?' ;
     wildcard = '*.*' ;
     DefaultDrive = 0 ;
begin
{$I-}
     name       :=  path_name ;
     path_save1 :=  null ;
     path_save2 :=  null ;

     GetDir (DefaultDrive, path_save1) ;

     if  pos(colon,name)=2  then
     begin
          drive := copy (name,1,2) ;
          ChDir (drive) ;
```

```
          if IoResult = NoError then GetDir (DefaultDrive,
path_save2) ;
     end ;


     ChDir ( name ) ;

     if  IoResult = NoError  then
         exist_path := true
     else
         exist_path := false
     {end if} ;

     if  path_save2 <> null  then ChDir (path_save2) ;
     if  path_save1 <> null  then ChDir (path_save1) ;
{$I+}
end ;

procedure  terminate_to_os_shell ;
var    i             : byte ;
const  screen_save : screen_type = nil ;
begin
     save_screen_to (screen_save)  ;
     write( 'Type EXIT to return to program...' ) ;
     wait;
     exec( 'C:\COMMAND.COM' , '' ) ;
     restore_screen_from (screen_save) ;
end ;


procedure  get_display_mode ( var  mode , page , col : byte
) ;
var  regs : registers ;
begin
     with regs do
     begin
          AH := $0F ;
          intr ( $10 , regs ) ;
          mode := AL ;
          col  := AH ;
          page:= BH ;
     end ;
end {get_display_mode} ;


procedure set_screen_ptr ( var scrptr : intptr ) ;
var  mode , page , col : byte ;
begin
     get_display_mode ( mode , page , col ) ;
     case  mode  of
         $02,$03 :  scrptr := ptr ( $B800 ,
(max_col*max_line) * page ) ;
           $07   :  scrptr := ptr ( $B000 ,
(max_col*max_line) * page ) ;
```

```
        else
                scrptr := ptr ( $B000 , $0000 ) ;
     end {case} ;
end {set_screen_ptr} ;

procedure  save_screen_to ( var screen_save : screen_type )
;
var
     screen  :  intptr ;
begin
     if  screen_save = nil  then
     begin
          GetMem ( screen_save , max_col*max_line*2) ;
          set_screen_ptr ( screen ) ;
          move ( screen^ , screen_save^ , max_col*max_line*2
) ;
     end {if} ;
end {save_screen} ;

procedure  restore_screen_from ( var screen_save :
screen_type ) ;
var
     screen  :  intptr ;
begin
     if  screen_save <> nil  then
     begin
          set_screen_ptr ( screen ) ;
          move ( screen_save^ , screen^ , max_col*max_line*2
) ;
          FreeMem ( screen_save , max_col*max_line*2) ;
          screen_save := nil ;
     end {if} ;
end {restore_screen} ;


(*

function  readkey : char ;
var
   key_pressed : boolean ;
   reg         : Registers ;

begin

{IFDEF MOUSE }
```

```
        else
                scrptr := ptr ( $B000 , $0000 ) ;
     end {case} ;
end {set_screen_ptr} ;

procedure  save_screen_to ( var screen_save : screen_type )
;
var
     screen  :  intptr ;
begin
     if  screen_save = nil  then
     begin
          GetMem ( screen_save , max_col*max_line*2) ;
          set_screen_ptr ( screen ) ;
          move ( screen^ , screen_save^ , max_col*max_line*2
) ;
     end {if} ;
end {save_screen} ;

procedure  restore_screen_from ( var screen_save :
screen_type ) ;
var
     screen  :  intptr ;
begin
     if  screen_save <> nil  then
     begin
          set_screen_ptr ( screen ) ;
          move ( screen_save^ , screen^ , max_col*max_line*2
) ;
          FreeMem ( screen_save , max_col*max_line*2) ;
          screen_save := nil ;
     end {if} ;
end {restore_screen} ;


(*

function  readkey : char ;
var
   key_pressed : boolean ;
   reg         : Registers ;

begin

{IFDEF MOUSE }
```

```
     repeat
           { get key status } reg.ah  :=  1 ; intr($16,reg)
;
           if  (reg.Flags and Fzero) = 0  { Flags zero clear
} then
              key_pressed := true
           else
              key_pressed := false
           {end if} ;

     until key_pressed or mouse_button_pressed ;

     if mouse_button_pressed then
          if left_button_pressed or right_button_pressed
then
               readkey := return
          else if middle_button_pressed then
               readkey := Esc
          {end if}
     else
{ENDIF}
        readkey := Crt.readkey ;
end ;


*)


procedure default_read ( var a : string ) ;
const
      null = ''  ;
      NUL  = #00 ;
      BS   = #08 ;
var
    c     : array [0..255] of char absolute a ;
    index : byte absolute a ;
    x,y   ,
    i     : byte ;
    inkey : char ;

begin
     write(a) ;

     inkey := readkey ;

     if  inkey = return  then exit ;
```

```
     if  (inkey = space) or (inkey = ESC)  then
     begin
          x := WhereX-index ; y := WhereY;
          gotoxy(x,y) ;
          for i:= 1 to index do write(space) ;
          a := null ;
          gotoxy(x,y) ;
     end {if} ;

     if  inkey = BS   then
     begin
          x := whereX-1 ; y := wherey ;
          gotoxy(x,y);
          write(space) ;
          gotoxy(x,y);
          c[index]:=NUL;
          dec(index);
     end ;

     if  inkey > space  then
     begin
          x := WhereX-index ; y := WhereY;
          gotoxy(x,y) ;
          for i:= 1 to index do write(space) ;
          a := inkey  ;
          gotoxy(x,y) ;
          write(a) ;
     end {if} ;

     repeat
          inkey := readkey ;
          case  inkey  of
              #32..#255  : begin
                              inc (index) ;
                              c[index] := inkey;
                              write(inkey) ;
                           end ;
                  esc    : begin
                              x := WhereX-index; y :=
WhereY;
                              gotoxy(x,y);
                              for i:= 1 to index do
write(space) ;
                              gotoxy(x,y);
                              a:=null;
                           end ;
```

```
                    BS      : if index > 0 then
                              begin
                                 x := whereX-1 ; y := wherey ;
                                 gotoxy(x,y);
                                 write(space) ;
                                 gotoxy(x,y);
                                 c[index]:=NUL;
                                 dec(index);
                              end ;
            end {case} ;
     until  inkey = return  ;
end ;



begin { init section }

end ..
```

ภาคผนวก ซ

ยูนิต complx

คำสั่งที่สร้างขึ้นในยูนิตนี้จะ เป็นคำสั่ง เกี่ยวกับการคำนวณจำนวนเชิงซ้อนซึ่งจำเป็นต้องใช้ในโปรแกรมนี้

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจดังนี้

1. ฟังชัน power มีพารามิเตอร์เป็นจำนวนจริงสองจำนวน ฟังชันนี้ใช้คำนวณการยกกำลังของเลขจำนวนจริงสองจำนวนนี้ แล้วส่งกลับผ่านชื่อของฟังชัน

2. ฟังชัน modulus มีพารามิเตอร์เป็นจำนวนเชิงซ้อน ฟังชันนี้จะคำนวณค่าสัมบูรณ์ของ จำนวนเชิงซ้อน แล้วส่งกลับผ่านชื่อของฟังชัน

3. ฟังชัน phase มีพารามิเตอร์เป็นจำนวนเชิงซ้อน ฟังชันนี้จะคำนวณเฟสของจำนวนเชิงซ้อน แล้วส่งกลับผ่านชื่อของฟังชัน

4. โพรซีเยอร์ swapc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสองจำนวน จะสลับที่ข้อมูลในพารามิเตอร์ทั้งสองแล้วส่งกลับผ่านพารามิเตอร์

5. โพรซีเยอร์ conj มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสองจำนวน จะคำนวณหาค่าคู่สังยุคของพารามิเตอร์ตัวแรก แล้วส่งกลับผ่านพารามิเตอร์ตัวที่สอง

6. โพรซิเยอร์ addc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสามจำนวน จะคำนวณหาผลบวกของพารามิเตอร์สองตัวแรก แล้วส่งผลบวกกลับผ่านพารามิเตอร์ตัวที่สาม

7. โพรซิเยอร์ subc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสามจำนวน จะคำนวณหาผลต่างของพารามิเตอร์สองตัวแรก แล้วส่งผลต่างกลับผ่านพารามิเตอร์ตัวที่สาม

8. โพรซิเยอร์ mulc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสามจำนวน จะคำนวณหาผลคูณของพารามิเตอร์สองตัวแรก แล้วส่งผลการคูณกลับผ่านพารามิเตอร์ตัวที่สาม

9. โพรซิเยอร์ divc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสามจำนวน จะคำนวณหาผลหารของพารามิเตอร์สองตัวแรก แล้วส่งผลหารกลับผ่านพารามิเตอร์ตัวที่สาม

10. โพรซิเยอร์ expc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสองจำนวน จะคำนวณหาค่าเอกซ์โปเนนเชียลของพารามิเตอร์ตัวแรก แล้วส่งผลกลับผ่านพารามิเตอร์ตัวที่สอง

11. โพรซิเยอร์ logc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสองจำนวน จะคำนวณหาค่าลอการิทึมของพารามิเตอร์ตัวแรก แล้วส่งผลกลับผ่านพารามิเตอร์ตัวที่สอง

12. โพรซิเยอร์ woperci มีพารามิเตอร์เป็นจำนวนเชิงซ้อนหนึ่งจำนวน และเป็นเลขจำนวนเต็มหนึ่งจำนวน จะคำนวณการยกกำลังของจำนวนทั้งสองนี้ แล้วส่งผลซึ่งเป็นจำนวนเชิงซ้อนกลับผ่านทางพารามิเตอร์ตัวที่สาม

13. โพรซิเยอร์ wopercr มีพารามิเตอร์เป็นจำนวนเชิงซ้อนหนึ่งจำนวน และเป็นเลขจำนวนจริงหนึ่งจำนวน จะคำนวณการยกกำลังของจำนวนทั้งสองนี้ แล้วส่งผลซึ่งเป็นจำนวนเชิงซ้อนกลับผ่านทางพารามิเตอร์ตัวที่สาม

14. โพรซิเยอร์ wopercc มีพารามิเตอร์เป็นจำนวนเชิงซ้อนสามจำนวน จะคำนวณการยกกำลังของจำนวนเชิงซ้อนสองตัวแรกนี้ แล้วส่งผลซึ่งเป็นจำนวนเชิงซ้อนกลับผ่าน

ทางพารามิเตอร์ตัวที่สาม

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไปแล้ว

```
unit  complx ;

(**                                    **)
(**    Filename   :  COMPLEX.PAS       **)
(**                                    **)

interface


type    real = double ;


const   two_pi   :  real   =  6.2831853072 ;
        pi_2     :  real   =  1.5707963268 ;

type    complex   =   record  case integer of
                              1 : ( re , im   :  real ) ;
                              2 : ( modu , pha   :  real ) ;
                              3 : ( general : array [0..1] of
real ) ;
                      end;


function  power( x,y : real ) : real ;

function  modulus ( c : complex ) : real ;
        { modulus is the magnitude of complex number }

function  phase ( c : complex ) : real ;

procedure  swapc( var x,y : complex ) ;

procedure  conj( x : complex ; var y : complex ) ;
             {  conjugate of x store to y  }

procedure  addc( x,y : complex ; var z : complex ) ;
        { add x to y store to z }


procedure  subc( x,y : complex ; var z : complex ) ;
        { substract y form x  store to z }

procedure  mulc( x,y : complex ; var z : complex ) ;
        { multiply  x to y store to z }

procedure  divc( x,y : complex ; var z : complex ) ;
        { divide  x by y   store to z }

procedure  expc( x : complex ; var y : complex ) ;

procedure  logc( x : complex ; var y : complex ) ;

procedure  powerci( x : complex ; i : integer ; var z :
complex ) ;
```

```
procedure  powercr( x : complex ; y : real ; var z : complex
) ;

procedure  powercc( x,y : complex ; var z : complex ) ;


(*************************************************************
******************)

implementation


function  power( x,y : real ) : real ;
begin
     power := exp(y*ln(x)) ;
end;


function  modulus ( c : complex ) : real ;
          { modulus is the magnitude of complex number }
begin
     modulus := sqrt( c.re*c.re + c.im*c.im ) ;
end ;


function  phase ( c : complex ) : real ;
var  slope : real ;
begin
     if (c.re = 0.0) and (c.im = 0.0) then
        slope := 0.0
     else
        if abs(c.im) < abs(c.re) then
           begin
                slope :=  arctan( c.im / c.re )  ;
                if c.re < 0.0 then slope :=  slope + pi ;
           end
        else
           begin
                slope :=  arctan( c.re / c.im )  ;
                if c.im < 0.0 then slope :=  slope + pi ;
                slope := pi_2 - slope ;
           end ;
        {end if}
     {end if}
     if slope < 0.0 then slope := two_pi + slope ;
     phase := slope ;
end ;
```

```
procedure  swapc( var x,y : complex ) ;
var  temp : complex ;
begin
     temp := x ;
     x := y ;
     y := temp ;
end ;


procedure  conj( x : complex ; var y : complex ) ;
             {  conjugate of x store to y  }
begin
     y.re := x.re ;
     y.im := - x.im ;
end;


procedure  addc( x,y : complex ; var z : complex ) ;
         { add x to y store to z }
begin
     z.re := x.re + y.re ;
     z.im := x.im + y.im ;
end;


procedure  subc( x,y : complex ; var z : complex ) ;
         { substract y form x  store to z }
begin
     z.re := x.re - y.re ;
     z.im := x.im - y.im ;
end;


procedure  mulc( x,y : complex ; var z : complex ) ;
         { multiply  x to y store to z }
begin
     z.re := x.re*y.re - x.im*y.im ;
     z.im := x.re*y.im + x.im*y.re ;
end;


procedure  divc( x,y : complex ; var z : complex ) ;
         { divide  x by y   store to z }
var temp : real ;
begin
     temp := y.re*y.re + y.im*y.im ;
     z.re := ( x.re*y.re + x.im*y.im ) / temp ;
     z.im := ( x.im*y.re - x.re*y.im ) / temp ;
end;
```

```
procedure  expc( x : complex ; var y : complex ) ;
begin
     y.re := exp(x.re) * cos(x.im) ;
     y.im := exp(x.re) * sin(x.im) ;
end;


procedure  logc( x : complex ; var y : complex ) ;
begin
     y.re := ln(modulus(x)) ;
     y.im := phase(x) ;
end;


procedure  powerci( x : complex ; i : integer ; var z :
complex ) ;
var  r,t  :  real    ;
     y    :  real    ;
begin
     y  :=  i ;    { convert integer i to real }
     r  :=  power ( modulus(x) , y ) ;
     t  :=  phase(x) * y ;
     z.re := r * cos(t) ;
     z.im := r * sin(t) ;
end;


procedure  powercr( x : complex ; y : real ; var z : complex
) ;
var  r,t  :  real ;
begin
     r  :=  power ( modulus(x) , y ) ;
     t  :=  phase(x) * y ;
     z.re := r * cos(t) ;
     z.im := r * sin(t) ;
end;



procedure  powercc( x,y : complex ; var z : complex ) ;
var  temp  :  complex ;
begin
     logc(x,temp) ;
     mulc(y,temp,temp) ;
     expc(temp,z) ;
end;


begin { init section }

end..
```

ภาคผนวก ฌ

ยูนิต contours

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการนำข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำ มาแสดงในรูปของเส้นลำดับขั้น

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจคือโพรซิเยอร์ draw_contour_in ซึ่งมีพารามิเตอร์สี่พารามิเตอร์ด้วยกันคือ

- ตัวแรกใช้กำหนดว่าจะแสดงเป็นเส้นระดับความสูงหรือใช้สีแทนระดับความสูงต่างๆ มีค่าได้สองค่าคือ curve และ color

- ตัวที่สองเป็นเมทริกซ์ที่ต้องการนำมาแสดงผล

- สองตัวสุดท้ายเป็นตำแหน่งบนจอภาพที่ต้องการให้นำภาพไปแสดงออก

นอกจากนี้แล้วยังมีตัวแปรที่น่าสนใจอีกสองตัว ตัวหนึ่งได้แก่ตัวแปรที่ใช้ในการกำหนดจำนวนเส้นระดับความสูงชื่อ contour_num ซึ่งจะมีค่ามากกว่าค่าคงที่ max_contour ไม่ได้ และอีกตัวหนึ่งเป็นตัวแปรที่ใช้กำหนดระดับต่ำสุดที่จะนำไปใช้ในการเขียนภาพชื่อ contour_base

โปรแกรมได้แสดงไว้ในหน้าถัดไป

```
unit contours ;

interface

uses  complx    , {unit of complex variable and function }
      crt,cell,
      vardef    , {variable definition unit}
      glue4     , {connect graphic command to simple command
}
      ploter    ,
      graph_2d ; { 2D graphics tools }


type
     contour_type  = (curve,color) ;

const
     max_contour   =  20     ;

     contour_color :  integer       =  2      ;
     contour_num   :  integer       =  6      ;
     contour_base  :  real          =  0.0    ;  { unit in %
of maximum peak }

procedure  ask_for_contour_num ;

procedure  ask_for_contour_base ;

procedure  draw_contour_in ( mode : contour_type ;
                             var u : matrix ;
                         {at} xo,yo : integer ) ;


implementation




var
     max_color      ,
     mosaic_scale   : integer  ;
     max_peak       : real     ;
     contour_level  : array [1..max_contour] of real ;


procedure  ask_for_contour_num ;
var
     answer : string ;
     temp   : integer ;
```

```
begin
     { *** Ask for number of contour line *** }
     writeln( ' Please enter numbers of contour .'     ) ;
     writeln( '  maximum  : ',max_contour:3,' lines') ;
     writeln( '  default  : ',contour_num:3,' lines') ;
     write  ( '>> ' ) ; readln (answer) ;
     if answer <> null then  val (answer,contour_num,temp) ;
end ;


procedure  ask_for_contour_base ;
var
     answer : string  ;
     temp   : integer ;
begin
     { *** Ask for contour base level *** }
     writeln( ' Please enter contour base level .'     ) ;
     writeln( '  range in 0-100 %') ;
     writeln( '  default  : ',100.0*contour_base:6:2,' %') ;
     write  ( '>> ' ) ; readln (answer) ;
     if answer <> null then
     begin
          val (answer,contour_base,temp) ;
          contour_base := contour_base / 100.00 ;
     end ;
end ;

procedure  curve_mosaic ( u1,u2,u3,u4  :  real    ;
                     {at}  x0,y0       :  integer ;
                   {size}  input_scale :  integer ) ;
var
     i     ,
     dot   : integer ;
     u5          ,
     temp        ,
     scale       ,
     max_level   ,
     min_level   : real ;
     x1,y1,x2,y2   : real ;
     xi1,yi1,xi2,yi2 : integer ;

          function max( x,y : real ) : real ;
          begin
               if x < y then  max := y  else  max := x ;
          end ;

          function min( x,y : real ) : real ;
          var  temp : real ;
          begin
               if x > y then  min := y  else  min := x ;
          end ;
```

```
        procedure solve_triangle ( u1,u2,u3 ,contour_level
: real ;
                                   var x1,y1,x2,y2 : real )
;
        var   a,b,c : real ;
              cross : byte ;
              x,y   : array [1..3] of real ;
              dot   : array [1..2] of byte ;
        begin
             c := u1 ;
             a := (u2-u1) / 2 ;
             b := u3 - a - c ;

             if  abs(a) > 1E-10  then x[1] :=
(contour_level-c) / a
               else x[1] := maximum_real ;
             y[1] := 0.0 ;

             if abs(a-b) > 1E-10 then x[2] :=
(contour_level-2*b-c) / (a-b)
               else x[2] := maximum_real ;
             y[2] := 2 - x[2] ;

             if abs(a+b) > 1e-10 then x[3] :=
(contour_level-c) / (a+b)
               else x[3] := maximum_real ;
             y[3] := x[3] ;

             cross := 1 ;

             if ( x[1] >= 0.0 ) and ( x[1] <= 2.0 ) then
               begin  dot[cross] := 1 ; cross := cross+1 ;
end ;
             if ( x[2] >= 1.0 ) and ( x[2] <= 2.0 ) then
               begin  dot[cross] := 2 ; cross := cross+1 ;
end ;
             if ( x[3] >= 0.0 ) and ( x[3] <= 1.0 ) then
               begin  dot[cross] := 3 ; cross := cross+1 ;
end ;

             x1 := x[ dot[1] ] * scale ;
             y1 := y[ dot[1] ] * scale ;
             x2 := x[ dot[2] ] * scale ;
             y2 := y[ dot[2] ] * scale ;
             if abs(x1) > maxint then x1 := maxint ;
             if abs(y1) > maxint then y1 := maxint ;
             if abs(x2) > maxint then x2 := maxint ;
             if abs(y2) > maxint then y2 := maxint ;
        end {solve_triangle} ;
```

```
begin
{$B-}
     u5 := ( u1+u2+u3+u4 ) / 4.0 ;
     scale := input_scale / 2.0 ;

     {high delta}
     max_level := max( max(u1,u2) , u5 ) ;
     min_level := min( min(u1,u2) , u5 ) ;
     for  i :=  1 to contour_num  do
     if ( min_level <= contour_level[i] ) and ( max_level >=
contour_level[i] )
     then
     begin
          solve_triangle ( u1,u2,u5 , contour_level[i] ,
x1,y1,x2,y2 ) ;
          xi1 := trunc( x1  + x0 ) ;
          yi1 := trunc( y1  + y0 ) ;
          xi2 := trunc( x2  + x0 ) ;
          yi2 := trunc( y2  + y0 ) ;
          draw (xi1,yi1,xi2,yi2 , contour_color ) ;
     end {if} ;

     {low delta}
     max_level := max( max(u3,u4) , u5 ) ;
     min_level := min( min(u3,u4) , u5 ) ;
     for  i :=  1 to contour_num  do
     if ( min_level <= contour_level[i] ) and ( max_level >=
contour_level[i] )
     then
     begin
          solve_triangle ( u4,u3,u5 , contour_level[i] ,
x1,y1,x2,y2 ) ;
          xi1 := trunc( x1 + x0 ) ;
          yi1 := trunc( - y1 + y0 + 2 * scale ) ;
          xi2 := trunc( x2 + x0 ) ;
          yi2 := trunc( - y2 + y0 + 2 * scale) ;
          draw (xi1,yi1,xi2,yi2 , contour_color ) ;
     end {if} ;

     {right delta}
     max_level := max( max(u3,u2) , u5 ) ;
     min_level := min( min(u3,u2) , u5 ) ;
     for  i :=  1 to contour_num  do
     if ( min_level <= contour_level[i] ) and ( max_level >=
contour_level[i] )
     then
     begin
          solve_triangle ( u2,u3,u5 , contour_level[i] ,
x1,y1,x2,y2 ) ;
          xi1 := trunc( -y1 + x0 + 2 * scale) ;
          yi1 := trunc( x1 + y0 ) ;
          xi2 := trunc( -y2 + x0 + 2 * scale ) ;
          yi2 := trunc( x2 + y0 ) ;
```

```
          draw (xi1,yi1,xi2,yi2 , contour_color ) ;
     end {if} ;

     {left delta}
     max_level := max( max(u1,u4) , u5 ) ;
     min_level := min( min(u1,u4) , u5 ) ;
     for  i :=  1 to contour_num  do
     if ( min_level <= contour_level[i] ) and ( max_level >=
contour_level[i] )
     then
     begin
          solve_triangle ( u1,u4,u5 , contour_level[i] ,
x1,y1,x2,y2 ) ;
          xi1 := trunc( y1 + x0 ) ;
          yi1 := trunc( x1 + y0 ) ;
          xi2 := trunc( y2 + x0 ) ;
          yi2 := trunc( x2 + y0 ) ;
          draw (xi1,yi1,xi2,yi2 , contour_color ) ;
     end {if} ;

end {of mosaic} ;



var  p,q  :  array [0..30] of real ;

          procedure declare_pq ( input_scale : integer ) ;
          var  i : integer ;
          begin
               for i := 0 to input_scale do
               begin
                    p[i] :=  i / input_scale ;
                    q[i] :=  p[i] ;
               end ;
          end ;



procedure color_mosaic  ( u1,u2,u3,u4  :  real    ;
                          {at}  x0,y0      :  integer ;
                         {size} input_scale  :  integer ) ;
var  x,y   :  real ;
     val   :  real ;
     temp  :  real ;
     base_level : real ;
     level :  integer ;
     i,j   :  integer ;
begin
     base_level := contour_base * max_peak ;
     temp := max_peak / max_color ;
```

```
     if temp < base_level 'then  temp := base_level ;
     if (u1<temp) and (u2<temp) and (u3<temp) and (u4<temp)
then exit ;

     temp  := max_color / (max_peak-base_level) ;
     for i := 0 to input_scale do
     begin
          x :=  u1-p[i]*(u1-u2) ;
          y :=  u4-p[i]*(u4-u3) ;

          for j := 0 to input_scale do
          begin
               val := x - q[j]*(x-y)  ;
               level := trunc ( temp * (val-base_level) ) ;
               if level > 0 then plot ( i+x0 , j+y0 , level
) ;
          end {j loop} ;

     end {i loop} ;
end {color_mosaic} ;


procedure draw_contour_in ( mode : contour_type ;
                            var u : matrix ;
                        {at} xo,yo : integer ) ;
var
    half_scale ,
    x,y    ,
    i,j    :  integer ;
    temp   :  real    ;
    base_level : real ;

begin { draw_contour  }

     {  AUTOMATIC SCALE  }

     mosaic_scale :=  GetMaxY div max_data ;

     if  mosaic_scale < 1  then  mosaic_scale  := 1    ;

     max_color  :=  GetMaxColor + 1 ;

    {  DRAW REFERANCE SHADE STRIP  }
     if mode = color then
     begin
          declare_pq ( mosaic_scale ) ;
```

```
          draw_color_strip_at (
xo+(mosaic_scale*(max_data))+10,
                                  yo+mosaic_scale*(max_data)-
max_color*16,
xo+(mosaic_scale*(max_data))+39,
                                  yo+mosaic_scale*(max_data)+1
,
                                  max_color ) ;
     end {if} ;


     max_peak :=  1.01 * max_peak_of_matrix( u ) ;

     {  CALCULATE CONTOUR LEVEL }

     base_level := contour_base * max_peak ;

     for i := 1 to contour_num  do
     begin

          contour_level [i] :=    i * (max_peak-base_level)
/ contour_num
                                  + base_level ;
     end;

     half_scale  :=  round (mosaic_scale / 2 ) ;

     open_graph_window( xo+half_scale,
                  yo+half_scale ,
                  xo+mosaic_scale*(max_data-1)+half_scale+2
,
                  yo+mosaic_scale*(max_data-1)+half_scale+2
) ;

     for j := max_data-1 downto 1 do
     for i := 0 to max_data-2  do
     begin
          x := i * mosaic_scale ; y := (max_data-j-1) *
mosaic_scale ;

       case mode of
```

```
            curve  :
            curve_mosaic (
                        sqrt( sqr(u^[i]^[j].re) +
sqr(u^[i]^[j].im) ) ,
                        sqrt( sqr(u^[i+1]^[j].re) +
sqr(u^[i+1]^[j].im) ) ,
                        sqrt( sqr(u^[i+1]^[j-1].re) +
sqr(u^[i+1]^[j-1].im) ) ,
                        sqrt( sqr(u^[i]^[j-1].re) +
sqr(u^[i]^[j-1].im) ) ,
                {at}    x+1,y+1 , mosaic_scale ) ;

            color :
            color_mosaic (
                        sqrt( sqr(u^[i]^[j].re) +
sqr(u^[i]^[j].im) ) ,
                        sqrt( sqr(u^[i+1]^[j].re) +
sqr(u^[i+1]^[j].im) ) ,
                        sqrt( sqr(u^[i+1]^[j-1].re) +
sqr(u^[i+1]^[j-1].im) ) ,
                        sqrt( sqr(u^[i]^[j-1].re) +
sqr(u^[i]^[j-1].im) ) ,
                {at}    x+1,y+1 , mosaic_scale ) ;

          end {case} ;

     end ;
     if  adeptor = plotter  then flush_plotter_buffer ;

end {plot_contour} ;


begin {init section}
end.
```

ภาคผนวก ฌ

ยูนิต display

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการนำข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำมาแสดงในเป็นภาพในลักษณะต่างๆตามแต่จะสั่งให้เป็นอย่างไร

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจคือ

1. โพรซิเยอร์ display_graph_of ซึ่งมีพารามิเตอร์ห้าพารามิเตอร์ด้วยกันคือ
   - พารามิเตอร์ตัวแรกเป็นเมทริกซ์ที่ต้องการนำมาแสดงผล
   - พารามิเตอร์ตัวที่สองใช้กำหนดว่าจะแสดงผลออกทางจอภาพในรูปแบบใด ดังนี้คือ
     - shade จะแสดงภาพเป็นจุดสี่เหลี่ยมสีต่างๆ
     - gray จะแสดงภาพโดยใช้ความหนาแน่นของจุด
     - contour จะแสดงภาพเป็นเส้นลำดับขั้น
     - block แสดงภาพโดยใช้จุดสี่เหลี่ยมขนาดต่างๆ
     - mix_shade เป็นการแสดงภาพผสมระหว่าง shade กับ contour
     - mix_gray เป็นการแสดงภาพผสมระหว่าง gray กับ contour
     - mix_block เป็นการแสดงภาพผสมระหว่าง block กับ contour
   - พารามิเตอร์ตัวที่สามใช้กำหนดเลือกชนิดของการแสดงผลในแบบเส้นลำดับขั้นว่าจะมีการแสดงจะแสดงเป็นเส้นระดับความสูงหรือใช้สีแทนระดับความสูงต่างๆ มีค่าได้สองค่าคือ curve และ color
   - พารามิเตอร์สองตัวสุดท้ายเป็นตำแหน่งบนจอภาพที่ต้องการให้นำภาพไป

แสดงออก

สำหรับการแสดงผลนี้ โปรแกรมจะแสดงผลออกให้มีขนาดใหญ่ที่สุดเท่าที่จอภาพจะสามารถทำได้เลยทีเดียวโดยที่ผู้ใช้ไม่ต้องจัดการในเรื่องนี้

2. โพรซิเยอร์ menu_and_display_graph_of จะทำการถามโดยใช้เมนูว่าต้องการให้มีการแสดงภาพในรูปแบบใด และจัดการแสดงภาพในรูปแบบที่ต้องการให้

```
unit display ;

interface

uses  dos,crt,graph,
      vardef    , {variable definition unit}
      glue4     , {connect graphic command to simple command
}
      cell      , {little tools}
      windsys   ,
      music     ,
      menu      , { menu manager}
      ploter    ,
      graph_2d  , { 2D graphics tools }
      massageg  ,
      contours  ; { contour graphics }


function    graph_mode ( item  :  byte  ) : paint ;

function    graph_contour_mode ( item  : byte  ) :
contour_type ;

procedure   display_graph_of ( var u : matrix ;
                        {in}   mode  : paint  ;
                               contour_mode : contour_type ;
                        {at}   x,y  : integer     ) ;

procedure  menu_and_display_graph_of ( var u : matrix )  ;

implementation

const
     null   = ''  ;
     esc    = #27 ;
     return = #13 ;

const
     graph_menu  :  menu_type = nil ;
     option_menu :  menu_type = nil ;
     graph_wind  :  window_type = nil ;
     infor_wind  :  window_type = nil ;
     option_wind :  window_type = nil ;
```

```
function  graph_mode ( item' : byte  ) : paint ;

begin
     case item  of
            1   : graph_mode := shade ;
            2   : graph_mode := gray  ;
            3   : graph_mode := block ;
            4   : graph_mode := contour ;
            5   : graph_mode := contour ;
            6   : graph_mode := mix_shade ;
            7   : graph_mode := mix_gray  ;
            8   : graph_mode := mix_block ;
          else
              graph_mode := block ;
     end {case} ;
end {graph_mode} ;


function  graph_contour_mode ( item  :  byte  ) :
contour_type ;
begin
     case item  of
            4   : graph_contour_mode := curve ;
            5   : graph_contour_mode := color ;
          else
              graph_contour_mode := color ;
     end {case} ;
end {graph_contour_mode} ;



procedure   display_graph_of ( var u : matrix ;
                         {in}   mode  : paint  ;
                                contour_mode : contour_type ;
                         {at}   x,y  : integer     ) ;
var
   answer    : string[80] ;
   temp      : integer    ;
   again     ,
   spooling  : boolean    ;
   inkey     : char       ;
   file_name : string   absolute   signal_data_file_name  ;
const
     spool_file  =  'spool.$$$' ;
```

```
const   dot_color_table : array [ paint , adeptor_type ] of
byte
                          =
((0,0,0,0),(2,3,1,2),(0,0,0,0),(2,3,1,2),
                            (0,0,0,0),(8,2,1,2),(8,2,1,2)) ;

    contour_color_table : array [ paint , adeptor_type ] of
byte
                          =
((0,0,0,0),(0,0,0,0),(2,3,1,2),(0,0,0,0),
                            (13,3,0,1),(2,2,1,3),(2,2,1,3)) ;

begin

     if  adeptor = plotter  then
     repeat
          cr;cr;
          writeln (' Do you want to use background plot
system . (y/n) ') ;cr;
          write   (' > ');
          inkey := readkey  ;
          case  inkey  of
               'Y','y' : begin
                              set_plotter_port_to (
spool_file ) ;
                              spooling := true ;
                              again := false ;
                              select_plotter ;
                         end ;
              'N','n' : begin
                              set_plotter_port_to ( 'lpt2' )
;
                              spooling := false ;
                              again := false ;
                              select_plotter ;
                         end ;
                  else
                      again := true ;
          end {case} ;
     until  not again ;

     graphcolormode;
     graphbackground(lightblue) ;
     palette(3) ;
```

```
     if  (adeptor = plotter) and spooling  then
     begin
          textmode ;
          clrscr ;
          clrscr ;
          writeln('- You select background plotting system')
;
          cr;
          write  ('- Your picture is writen to file
"'+spool_file+'"') ;
          write  (' and shall be dump to plotter while
you');
          writeln(' work with another job .') ;
          HighVideo;cr;
          writeln(' It is necessary to delete file
"'+spool_file+'"') ;
          writeln(' after your picture is finish .') ;cr;
          writeln('                    !WAIT!') ;
          LowVideo ;

     end {if};

     contour_color   := contour_color_table [mode,adeptor] ;
     gray_dot_color  := dot_color_table [mode,adeptor] ;
     block_dot_color := dot_color_table [mode,adeptor] ;


     {draw massage on graphics}
     case  mode  of
       shade,mix_shade : display_massage_on ( 480,10 ,
635,340 ,file_name) ;
       contour         : if contour_mode = color then
                           display_massage_on (480,10,
635,340 ,file_name)
                         else
                           display_massage_on (460,10,
635,340 ,file_name)
                         {end if} ;
     else
         display_massage_on (460,10,635,340 ,file_name );
     end {case} ;

     case  mode  of
          shade : begin
                       display_power_spectrum_in( shade ,u,
x,y ) ;
                  end ;
           gray : begin
                       display_power_spectrum_in( gray ,u,
x,y ) ;
                  end;
```

```
            block : begin
                        display_power_spectrum_in( block ,u,
x,y ) ;
                    end;
          contour : begin
                        draw_contour_in ( contour_mode ,u,
x,y ) ;
                    end ;
        mix_shade : begin
                        display_power_spectrum_in( shade ,u,
x,y ) ;
                        draw_contour_in ( curve ,u, x,y ) ;
                    end ;
         mix_gray : begin
                        display_power_spectrum_in( gray ,u,
x,y ) ;
                        draw_contour_in ( curve ,u, x,y ) ;
                    end ;
        mix_block : begin
                        display_power_spectrum_in( block ,u,
x,y ) ;
                        draw_contour_in ( curve ,u, x,y ) ;
                    end ;
      end {case} ;

      if  (adeptor = plotter) and spooling  then
      begin
           spooling_file ( spool_file {to}, 'LPT2' ) ;
           clrscr ;

           clrscr ;
           writeln('- You select background plotting system')
;
           cr;
           write  ('- Your picture is writen to file
"'+spool_file+'"') ;
           write  (' and shall be dump to plotter while
you');
           writeln(' work with another job .') ;
           HighVideo;cr;
           writeln(' It is necessary to delete file
"'+spool_file+'"') ;
           writeln(' after your picture is finish .') ; cr;
           writeln('          PRESS ANY KEY TO CONTINUE') ;
           LowVideo ;

      end {if};

      music_call; wait;

      textmode ;
      clrscr;

end {display_graph_of}  ;
```

```
procedure  menu_option ;
var
     item  : byte   ;
     inkey : char   ;
     numStr   : string ;
     cbaseStr : string ;
     gbaseStr : string ;
begin
     open_window ( option_wind ) ;clrscr ;
     move_window_to ( 2,9 ,relative ) ;

     repeat
           new_menu (option_menu) ;

           str ( contour_num   : 3   , numStr ) ;
           str ( 100*contour_base  :6:2  , cbaseStr ) ;
           str ( 100*graph_2d_base :6:2  , gbaseStr ) ;

       add_menu( option_menu ,'G','     Graph   base at
'+gbaseStr+' %      ');
       add_menu( option_menu ,'C','     Contour base at
'+cbaseStr+' %      ');
       add_menu( option_menu ,'O','     cOntour       :
'+numStr+' lines      ');

           add_menu( option_menu ,'E','     Exit.');

           clrscr ;

           select_menu ( option_menu ,item ,inkey ,{at} 1,1
) ;

        if  (inkey = return) and (item < 4)   then
        begin
          open_window (infor_wind) ; clrscr;cr;cr;
                move_window_to ( 0,0 ,relative ) ;

                case  item  of
                        1  :  ask_for_graph_2d_base ;
                        2  :  ask_for_contour_base ;
                        3  :  ask_for_contour_num ;
                end {case} ;

                move_window_to (1,1 , of_screen) ;
                close_window ;
           end {if} ;

     until (inkey = esc) or ((inkey = return) and (item =
4))  ;
     move_window_to (1,1 ,of_screen) ;
     close_window ;
end;
```

```
procedure  menu_and_display_graph_of ( var u : matrix )  ;
var
     item  : byte   ;
     inkey : char   ;
     mode  : paint   ;
     contour_mode : contour_type ;
begin
     open_window ( graph_wind ) ;

     repeat
           reset_max_peak ;

           textmode ;
           clrscr ;

           select_menu ( graph_menu ,item ,inkey ,{at} 1,1 )
;
           if  inkey = return  then
             case  item  of
                   1..8 :  begin
                                mode := graph_mode (item) ;
                                contour_mode :=
graph_contour_mode (item) ;
                                display_graph_of ( u {in}
,mode,contour_mode,
                                                   {at}  10,10) ;
                           end ;
                     9  :  menu_option ;
             end {case}
           {end if}

     until  (inkey = esc) or ((inkey = return) and (item =
10))
            or (last_menu_scan_key=RightArrow)
            or (last_menu_scan_key=LeftArrow) ;

     close_window { graph_wind } ;
end;


begin {init section}

     define_window  ( graph_wind  , 33,10, 62,21 ) ;
     window_color   ( graph_wind  , LightGray , Blue ) ;
     broader_color  ( graph_wind  , Yellow , Green ) ;
     window_heading ( graph_wind  , ' GRAPHICS DISPLAY ') ;
     head_color     ( graph_wind  , Blue , CYan ) ;
```

```
     define_window  ( option_wind , 1,1, 38,6 ) ;
     window_color   ( option_wind , LightGray , Blue ) ;
     broader_color  ( option_wind , Yellow , Green ) ;
     window_heading ( option_wind , ' GRAPHICS DISPLAY ') ;
     head_color     ( option_wind , Blue , CYan ) ;

     define_window  ( infor_wind  , 1,1 , 38,6 ) ;
     window_color   ( infor_wind  , White , Blue ) ;
     broader_color  ( infor_wind  , LightGreen , Green ) ;
     window_heading ( infor_wind  , ' OPTION ') ;
     head_color     ( infor_wind  , Blue , CYan ) ;


     new_menu (graph_menu) ;

     menu_normal_color    ( graph_menu , Yellow ,Blue ) ;
     menu_inverse_color   ( graph_menu , White , Magenta ) ;
     menu_highlight_color ( graph_menu , LightCyan ,Blue ) ;

     add_menu( graph_menu ,'C',' Color block ');
     add_menu( graph_menu ,'R',' Random dot ');
     add_menu( graph_menu ,'G',' Gray block ');
     add_menu( graph_menu ,'L',' Line contour ');
     add_menu( graph_menu ,'O',' cOlor contour ');
     add_menu( graph_menu ,'B',' color Block & contour line
');
     add_menu( graph_menu ,'A',' rAndom dot  & contour line
');
     add_menu( graph_menu ,'Y',' graY block  & contour line
');
     add_menu( graph_menu ,'P',' oPtion ');
     add_menu( graph_menu ,'E',' Exit to main menu . ');


     new_menu (option_menu) ;

     menu_normal_color    ( option_menu , Yellow ,Blue ) ;
     menu_inverse_color   ( option_menu , White , Magenta )
;
     menu_highlight_color ( option_menu , LightCyan ,Blue )
;

end..
```

ภาคผนวก ฎ

ยูนิต fourier

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการทำการแปลงข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำโดยวิธีการแปลงแบบฟูเรียร์อย่างเร็ว

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่เรียกใช้ได้เพียงคำสั่งเดียวคือคำสั่ง transform_matrix ซึ่งจะมีพารามิเตอร์เป็นเมทริกซ์ที่ต้องการนำมาแปลง โดยที่โพรซิเยอร์นี้จะแสดงเมนูขึ้นมาถามว่าต้องการทำการแปลงธรรมดา หรือการแปลงแบบผกผัน แล้วจึงจะไปทำการแปลงให้เป็นที่เรียบร้อย

สำหรับโปรแกรมได้แสดงไว้ในหน้าต่อไปแล้ว

```
unit  fourier ;

interface

uses  crt       ,
      menu      ,
      complx    , {unit of complex variable and function }
      vardef    , {variable definition unit}
      cell      , {little tool}
      windsys   ,
      music     ,
      timing    ;

procedure  transform_matrix ( var u : matrix ) ;

implementation


var
     w          :  vector  ;
     half_data  :  integer ;

const
     transform_menu : menu_type = nil ;
     transform_wind : window_type = nil ;
     infor_wind     : window_type = nil ;
     esc    = #27;
     return = #13;

function  bit_reverse ( source , digit : integer ) : integer
;

var    i       ,
       orbit   :   integer ;
begin
     orbit  :=  0 ;
     for  i := 1 to digit do
     begin
          orbit := ( source and 1 ) or orbit ;
          orbit := orbit shl 1 ;
          source := source shr 1 ;
     end {for} ;
     bit_reverse := orbit shr 1 ;
end {function bit_reverse} ;


function  two_power ( i : integer ) : integer ;
begin
     two_power :=  1 shl i ;
end {function two-power} ;
```

```
procedure  create_table_of ( var w : vector ) ;
var     i     :  integer ;
        x     ,
        d     ,
        c     :  real ;
begin
     x :=  0.0 ;
     d :=  sqrt( max_data ) ;
     c :=  2 * pi / max_data ;
     for i := 0 to max_data do
     begin
       w[i].re  := cos( x )  ;
       w[i].im  := sin( x )  ;
          x := x + c ;
     end ;
end {procedure  create_table_of} ;

procedure inverse_table ( var w : vector ) ;
var   i : integer  ;
begin
     for  i := 0 to max_data  do   w[i].im := -w[i].im ;
end ;


function log2( x : integer ) : integer ;
var  y  :  real ;
begin
     y  := x ;
     log2 := round( ln(y)/ln(2.0) ) ;
end;


procedure  fft ( var g : vector ) ;

           procedure swap ( var x,y : integer ) ;
           var temp : integer ;
           begin
                temp := x ;  x := y ;  y := temp ;
           end ;

var
        omega          :  real    ;
        temp           ,
        jet            ,
        source,target  ,
        max_j,max_k    ,
        p,q,t,l,j,k    :  integer ;
        dummy          :  complex  ;
        f              :  array [0..1] of vector ;
```

```
begin  { fast fourier transform }

      t  :=  log2(max_data) ;
         half_data := max_data div 2 ;

      target  := 1 ;
      source  := 0 ;

      f[source] :=  g  ;

      {  swap  right-half with left_half  }
      for  j := 0 to half_data-1 do
        swapc( f[source][j] , f[source][j+half_data] ) ;

      for  j := 0 to max_data-1 do
        f[target][j ] := f[source][bit_reverse(j,t) ] ;

      swap( source, target) ;

      for  l  :=   1  to  t  do
      begin
           max_j  :=  two_power(l-1) - 1 ;
           max_k  :=  two_power(t-l) - 1 ;
           for  j  :=  0  to  max_j  do
           begin
                for  k  :=  0  to  max_k  do
                begin
                     p := j + k * ( 1 shl l ) ;
                     q := p + ( 1 shl (l-1) ) ;

                     { multiply }
                     temp := 1 shl (t-l) ;
                     jet := j*temp ;
               dummy.re :=    f[source][q].re * w[jet].re
                      - f[source][q].im * w[jet].im ;
               dummy.im :=    f[source][q].re * w[jet].im
                      + f[source][q].im * w[jet].re ;

                     { addition }
                     f[target][p].re :=  f[source][p].re +
dummy.re ;
                     f[target][p].im :=  f[source][p].im +
dummy.im ;

                     { substract }
                     f[target][q].re :=  f[source][p].re -
dummy.re ;
                     f[target][q].im :=  f[source][p].im -
dummy.im ;

                end {for} ;
           end {for} ;
           swap( source,target ) ;
      end {for} ;
```

```
      {  swap  right-half with left_half  }
      for  j := 0 to half_data-1 do
        swapc( f[source][j] , f[source][j+half_data] ) ;

      g := f[source] ;

end {fft} ;

procedure  transpost ( var u : matrix ) ;
var
       i,j   :  integer ;
       temp  :  complex ;
begin
      for  i := 0 to max_data do
      for  j  :=  i+1 to max_data  do
      begin
        temp      :=  u^[i]^[j] ;
        u^[i]^[j] :=  u^[j]^[i] ;
        u^[j]^[i] :=  temp ;
      end;
end {transpost} ;

procedure two_dim_fft( var u : matrix ) ;

var    i       :  integer ;

begin
      cr;
      for  i  :=   0  to  max_data-1 do
      begin
           write ( #13,' tranform row : ',i ) ;
        fft ( u^[i]^ ) ;
      end ;
      cr;

      writeln (' transpost matrix ' ) ;
      transpost ( u ) ;
      cr;

      for  i  :=   0  to  max_data-1 do
      begin
           write ( #13,' tranform col : ',i ) ;
        fft ( u^[i]^ ) ;
      end ;
      cr;

      writeln (' transpost matrix ' ) ;
      transpost ( u ) ;
      cr;
end ;
```

```
procedure inverse_fft ( var u : vector ) ;
var i : integer ;
begin
     inverse_table ( w ) ;
     fft ( u ) ;
     inverse_table ( w ) ;

     for i :=  0 to max_data-1 do
     begin
       u[i].re := u[i].re / max_data ;
       u[i].im := u[i].im / max_data ;
     end ;
end {inverse_fft};


procedure inverse_2dfft ( var u : matrix ) ;
var i,j : integer ;
    d   : real ;
begin

     inverse_table ( w ) ;
     two_dim_fft ( u ) ;
     inverse_table ( w ) ;

     d :=  sqr(max_data) ;
     for i :=  0 to max_data-1 do
     for j :=  0 to max_data-1 do
     begin
       u^[i]^[j].re := u^[i]^[j].re / d ;
       u^[i]^[j].im := u^[i]^[j].im / d ;
     end ;
end ;


procedure transform_matrix ( var u : matrix ) ;

var  again : boolean ;
     do_transform : boolean ;
     item  : byte ;
     inkey : char ;
begin

     open_window ( transform_wind ) ;

     repeat
           clrscr ;
           gotoxy(3,15) ;

           select_menu ( transform_menu ,item ,inkey ,{at}
1,1 ) ;
```

```
            again := not ((inkey = esc) or ((inkey = return)
and (item = 3))
                          or (last_menu_scan_key=RightArrow)
                          or (last_menu_scan_key=LeftArrow)
);
            do_transform := again ;


            if  do_transform  then
            begin
                 case  item  of
                        1   : window_heading (infor_wind,'
TRANSFORM ');
                        2   : window_heading (infor_wind,'
INVERSE TRANSFORM ');
                 end {case} ;
                 open_window ( infor_wind ) ;
                 clrscr ; cr;
                 move_window_to ( 2,2, relative ) ;

                 start ;

                 case  item  of
                         1   :  two_dim_fft (u) ;
                         2   :  inverse_2dfft (u) ;
                 end {case} ;

                 cr;
                 writeln (' End transform .') ; cr;
                 stop ;
                 writeln (' Time used : ',base_60(time_used))
;
                 music_call;
                 wait ;
                 clrscr ;
                 move_window_to (1,1 , of_screen ) ;
                 close_window ;
            end {if} ;

     until  not again ;

     close_window ;

end ;
```

```
begin {initialize section }

     create_table_of ( w ) ;

     new_menu ( transform_menu ) ;
     menu_normal_color    ( transform_menu , Yellow ,Blue )
;
     menu_inverse_color   ( transform_menu , White , Magenta
) ;
     menu_highlight_color ( transform_menu , LightCyan ,Blue
) ;

     add_menu ( transform_menu ,'F',' Fast Fourier Transform
') ;
     add_menu ( transform_menu ,'I',' Inverse Fast Fourier
Transform ') ;
     add_menu ( transform_menu ,'E',' Exit to main menu .')
;

     define_window  ( transform_wind , 18,10 , 51,14 ) ;
     window_color   ( transform_wind , LightGray , Blue ) ;
     broader_color  ( transform_wind , White , Cyan ) ;
     window_heading ( transform_wind ,
                      ' FAST FOURIER TRANSFORMS ') ;
     head_color     ( transform_wind , Blue , Cyan ) ;

     define_window  ( infor_wind , 1,1 , 34,6 ) ;
     window_color   ( infor_wind , White , Blue ) ;
     broader_color  ( infor_wind , Yellow , Cyan ) ;
     window_heading ( infor_wind , ' TRANSFORM ') ;
     window_heading ( infor_wind , ' INVERSE TRANSFORM ') ;
     head_color     ( infor_wind , Blue , Cyan ) ;

end..
```

ภาคผนวก ฎ

ยูนิต getdata

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการนำข้อมูลที่อยู่ในเมทริกซ์ในหน่วยความจำบันทึกลงในแผ่นบันทึกข้อมูล หรืออ่านข้อมูลจากแผ่นบันทึกข้อมูลเข้ามาเก็บไว้ในเมทริกซ์ในหน่วยความจำ

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจคือ file_data_to มีพารามิเตอร์เป็นเมทริกซ์ โพรซิเยอร์นี้ จะแสดงเมนูออกมาถามว่าต้องการอ่านหรือเขียนข้อมูลในแฟ้มข้อมูลใด แล้วจึงจัดการให้เป็นไปตามที่ผู้ใช้ประสงค์

โปรแกรมได้แสดงไว้ในหน้าต่อไป

```
unit getdata ;

interface

uses  dos       ,
      crt       ,
      complx    , {unit of complex variable and function }
      vardef    , {variable definition unit}
      windsys   ,
      music     ,
      menu      ,
      timing    ,
      cell      ,  {little tools}
      askFname  ;  { file name select }

procedure  retrive_signal_to ( var u : matrix ) ;

procedure  store_signal_from ( var u : matrix ) ;

procedure  file_data_to ( var u : matrix ) ;

implementation

const
        getdata_wind  : window_type = nil  ;
        file_wind     : window_type = nil  ;
        file_menu     : menu_type   = nil  ;

        max_pos       : integer     = 80 ;


procedure  retrive_data ( var u : matrix ) ;
var
       i,j   :  integer ;
       temp ,
       x,y   :  real    ;
       z     :  complex ;

       data_file  :  file of complex  ;

begin
     assign ( data_file , signal_data_file_name ) ;
     reset  ( data_file ) ;
     for j :=  0  to  max_data-1  do
     begin
        for  i  :=  0 to max_data-1  do   read( data_file ,
u^[i]^[j] ) ;
     end {for} ;
```

```
     close ( data_file ) ;
end;


procedure  store_data ( var u : matrix ) ;
var
       i,j  :  integer ;
       temp ,
       x,y  :  real    ;
       z    :  complex ;

       data_file  :  file of complex  ;

begin
     assign ( data_file , signal_data_file_name ) ;
     rewrite( data_file ) ;
     for j :=  0  to  max_data-1  do
     begin
          for  i  :=  0 to max_data-1  do  write( data_file
, u^[i]^[j] ) ;
     end {for} ;
     close ( data_file ) ;
end;



function  file_is_data_file : boolean ;
var
       data_file  :  file of complex  ;

begin
     assign ( data_file , signal_data_file_name ) ;
     reset  ( data_file ) ;

     if  FileSize (data_file) >=  sqr(max_data)  then
         file_is_data_file  :=  true
     else
         file_is_data_file  :=  false
     {end if} ;
     close  ( data_file ) ;
end {of function} ;


procedure  retrive_signal_to ( var u : matrix ) ;
var
   again     : boolean ;
   name_save : string ;
begin
     name_save := signal_data_file_name ;
```

```
     repeat
          writeln( '     Please enter your file name OR ');
          writeln( '  <Space> & <Enter> key for directory'
) ;

          ask_for_file_name ( signal_data_file_name , 4,17
) ;

          clrscr ;

          if exist_file ( signal_data_file_name ) then
          begin
              if  file_is_data_file  then
                begin
                   cr;
                   writeln (' Retrive data from file " ',
                              signal_data_file_name,' " .')
;
                   start;
                   retrive_data (u) ;
                   stop ;
                   writeln(' Time used :
',base_60(time_used)) ;
                   music_call;
                   again := false ;
                end
              else
                begin
                   writeln (' File "
',signal_data_file_name,
                            ' " isn''t data file .') ;
                   writeln( '   *** TRY AGAIN ***    ',bell
) ;
                   again := true ;
                   signal_data_file_name := name_save ;
                   wait
                end
              {end if} ;
          end {if}
          else
          begin
              if signal_data_file_name <> ''  then
                begin
                   writeln( ' File  "
',signal_data_file_name,
                            ' " NOT founded. ') ;
                   writeln( '   *** TRY AGAIN ***    ',bell
) ;
                   again := true ;
                   signal_data_file_name := name_save ;
                   wait
                end
              else
```

```
                begin
                   cr;
                   writeln( ' You have not select any files
.');
                   signal_data_file_name := name_save ;
                   again := false ;
                   waiter;
                end
             {end if} ;
          end
          {end if} ;

     until  not again  ;


end {retrive_signal_to};


procedure  store_signal_from ( var u : matrix ) ;
var
   name_save : string ;
   inkey     : char   ;
begin
     name_save := signal_data_file_name ;

     writeln( '     Please enter your file name OR ');
     writeln( '  <Space> & <Enter> key for directory' ) ;

     ask_for_file_name ( signal_data_file_name , 4,17  ) ;
     clrscr ;

     if exist_file ( signal_data_file_name ) then
       begin
          writeln ( ' file ',signal_data_file_name,' already
exist .') ;
          writeln ( ' Overwrite ? (Y/N) ');
          repeat  inkey := readkey  until inkey in
['y','Y','n','N']  ;

          if (inkey = 'y') or (inkey = 'Y')  then
            begin
               writeln( ' File  " ',signal_data_file_name,'
" is save. ') ;
               store_data (u) ;
            end
          else
            begin
               signal_data_file_name := name_save ;
            end
          {end if}

       end
     else
```

```
        begin
           if signal_data_file_name <> ''  then
             begin
                writeln( ' File  " ',signal_data_file_name,'
" is save. ') ;
                store_data (u) ;
             end
           else
             begin
                cr;
                writeln( ' You have not select any files .');
                cr;
                writeln( ' your data isn''t save ') ;
                signal_data_file_name := name_save ;
                waiter;
             end
           {end if} ;
         end
        {end if} ;

end {store_signal_from};


procedure  file_data_to ( var u : matrix ) ;

var  again : boolean ;
     item  : byte ;
     inkey : char ;
begin

     open_window ( file_wind ) ;
     clrscr ;

     repeat
           select_menu ( file_menu ,item ,inkey ,{at} 1,1 )
;

           if  inkey = return  then
           begin
                open_window ( getdata_wind ) ; clrscr ;
                move_window_to ( 2,2 ,relative ) ;

                case item  of
                       1  :  retrive_signal_to ( u ) ;
                       2  :  Store_signal_from ( u ) ;
                end ;
                move_window_to ( 1,1 , of_screen ) ;
                close_window ;
           end {if} ;
```

```
          again := not ((inkey = esc) or ((inkey = return)
and (item = 3))
                        or (last_menu_scan_key=RightArrow)
                        or (last_menu_scan_key=LeftArrow)
);
     until  not again ;

     close_window ;

end ;



begin
     define_window  ( getdata_wind , 1,1 , 58,5 ) ;
     window_color   ( getdata_wind , Yellow , Blue ) ;
     broader_color  ( getdata_wind , LightCyan , LightGray )
;
     window_heading ( getdata_wind , ' Retrive data. ') ;
     head_color     ( getdata_wind , White , Cyan ) ;

     define_window  ( file_wind , 1,11 , 23,15 ) ;
     window_color   ( file_wind , Yellow , Blue ) ;
     broader_color  ( file_wind , LightCyan , Black ) ;
     window_heading ( file_wind , ' File ') ;
     head_color     ( file_wind , White , Cyan ) ;

     new_menu ( file_menu ) ;
     menu_normal_color    ( file_menu , Yellow ,Blue ) ;
     menu_inverse_color   ( file_menu , White , Magenta ) ;
     menu_highlight_color ( file_menu , LightCyan ,Blue ) ;

     add_menu ( file_menu ,'R',' Read data from file ') ;
     add_menu ( file_menu ,'S',' Save data to file   ') ;
     add_menu ( file_menu ,'E',' Exit to main menu   ') ;
end..
```

ภาคผนวก ฐ

ยูนิต glue4

เนื่องจากในการพัฒนาโปรแกรมนี้มีการใช้คำสั่งทางด้านกราฟฟิกเป็นจำนวนมาก เพื่อความสะดวกในการเขียนโปรแกรม จึงได้สร้างคำสั่งที่จะช่วยในการใช้งานต้นนี้ขึ้นมา ซึ่งก็ได้แก่คำสั่งที่สร้างขึ้นในยูนิตนี้

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจดังนี้

1. โพรซิเยอร์ plot จะเขียนจุดลง ณ ตำแหน่งที่กำหนดโดยพารามิเตอร์

2. โพรซิเยอร์ line มีพารามิเตอร์สี่ตัวด้วยกัน ซึ่งจะกำหนดจุดสองจุดที่จะลากเส้นเชื่อมจุดทั้งสองนี้

3. โพรซิเยอร์ graphwindow มีพารามิเตอร์สี่ตัวด้วยกัน ซึ่งจะกำหนดจุดสองจุดที่จะให้เป็นกรอบของหน้าต่างที่จะใช้งานในแบบกราฟฟิก

4. โพรซิเยอร์ InitGraphic ไม่มีพารามิเตอร์ ใช้สั่งให้โปรแกรมเตรียมตัวที่จะทำงานด้านกราฟฟิก

5. โพรซิเยอร์ graphcolormode ไม่มีพารามิเตอร์ ใช้สั่งให้เปลี่ยนการแสดงผลบนจอภาพเป็นแบบกราฟฟิก

6. โพรซิเยอร์ hires ไม่มีพารามิเตอร์ ใช้สั่งให้เปลี่ยนการแสดงผลบนจอภาพเป็นแบบกราฟฟิก ซึ่งมีการใช้สีต่างไปจากคำสั่ง graphcolormode

7. โพรซีเยอร์ textmode ไม่มีพารามิเตอร์ ใช้สั่งให้เปลี่ยนการแสดงผลบนจอภาพจากแบบกราฟฟิกกลับมาเป็นแบบตัวอักษรธรรมดา

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไปแล้ว

```
unit  glue4 ;

interface {section}

uses  crt,graph,cell ;


type
      adeptor_type  = (EGA,CGA,Hercules,plotter) ;

const
      adeptor           : adeptor_type = EGA ;


{$IFNDEF VDISK}

  {$IFDEF  VER50}
     BGI_path     =    'c:\tp'   ;
  {$ENDIF}

  {$IFDEF  VER40}
     BGI_path     =    'c:\turbo_p4'   ;
  {$ENDIF}

{$ELSE}
     BGI_path     =    ''   ;
{$ENDIF}



var
      aspect_ratio      : real ;
      background_color : byte ; {used for text}
      foreground_color : byte ; {used for text}


procedure  plot( x,y , color : integer ) ;

procedure  draw( x1,y1,x2,y2 , color : integer ) ;

procedure  graphwindow( x1,y1,x2,y2 : integer) ;

procedure  InitGraphic ;

procedure  graphcolormode ;

procedure  hires ;

procedure  textmode ;

procedure  palette ( n : integer ) ;

procedure  graphbackground ( n : integer ) ;
```

```
procedure  hirescolor ( n : integer ) ;

(*************************************************************
***************)

implementation {section}


procedure  plot( x,y , color : integer ) ;
begin
     PutPixel ( trunc(x*aspect_ratio),y ,color ) ;
     PutPixel ( round(x*aspect_ratio),y ,color ) ;
end ;

procedure draw( x1,y1,x2,y2 , color : integer ) ;
begin
     SetColor (color) ;
     SetLineStyle ( SolidLn ,0, NormWidth ) ;
     line ( round(x1*aspect_ratio),y1 ,
round(x2*aspect_ratio),y2 ) ;
end ;

procedure graphwindow( x1,y1,x2,y2 : integer) ;
begin
     SetViewPort( round(x1*aspect_ratio),y1 ,
                  round(x2*aspect_ratio),y2 ,
                  ClipOn ) ;
end ;

procedure InitGraphic ;
var
   Xasp      ,
   Yasp      : word ;
   grdriver ,
   grmode    : integer ;
begin
     grDriver :=  Detect ;
     grmode   :=  Detect ;
     initgraph ( grdriver , grmode ,BGI_path  ) ;

     if  adeptor in [CGA,EGA,Hercules]  then
     case  grdriver  of
               1  :  adeptor := CGA ;
               3  :  adeptor := EGA ;
               7  :  adeptor := Hercules ;
     end {case} ;

     GetAspectRatio( Xasp,Yasp ) ;
     aspect_ratio  :=  Yasp / Xasp ;
     if  adeptor = CGA  then  aspect_ratio := 1 ;

     RestoreCrtMode ;
end ;
```

```
const   screen_save : screen_type = nil ;

procedure  graphcolormode ;
var
   i         ,
   grmode    : integer ;

const  col_no : array [0..15] of integer  =
(*                 (7,8,1,9,13,5,12,4,2,10,3,6,14,11,15,0)
;(**)

(*               (63,8,56,88,1,17,19,9,57,25,11,27,31,63,0,0)
;(**)


(7,8,1,9,25,57,43,91,19,58,30,62,38,116,36,68);     (**)

begin
     save_screen_to (screen_save) ;

     case  adeptor  of
             CGA     :  grMode := 1 ;
             EGA     :  grMode := 1 ;
         Hercules    :  grMode := 0 ;
         plotter     :  grMode := 1 ;
     end {case} ;
     SetGraphMode (grMode) ;
     SetViewPort( 0,0 ,GetMaxX,GetMaxY , ClipOn ) ;
     if  adeptor = EGA  then  for i := 0 to 15 do
SetPalette(i,col_no[i]) ;

end;

procedure  hires ;
var
     i         ,
     grmode    : integer ;
begin
     save_screen_to (screen_save) ;
     case  adeptor  of
             CGA     :  grMode := 1 ;
             EGA     :  grMode := 1 ;
         Hercules    :  grMode := 0 ;
         plotter     :  grMode := 1 ;
     end {case} ;
     SetGraphMode (grMode) ;
     SetViewPort( 0,0 ,GetMaxX,GetMaxY , ClipOn ) ;
end;
```

```
procedure  textmode ;
begin
     RestoreCrtMode ;
     restore_screen_from (screen_save) ;
end ;

procedure  palette ( n : integer ) ;
begin
     { dummy }
end;


procedure  graphbackground ( n : integer ) ;
begin
     { dummy }
end ;

procedure  hirescolor ( n : integer ) ;
begin
     { dummy }
end ;


begin  { initialize section }
     background_color := Blue      ;
     foreground_color := LightGray ;
     clrscr;

     InitGraphic ;

     {set text color}

     textmode ;
end .
```

ภาคผนวก ฉ

ยูนิต graph_2d

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในการนำข้อมูลในเมทริกซ์ออกมาแสดงบนจอภาพในลักษณะต่างๆตามแต่จะกำหนด

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจคือโพรซิเยอร์ display_power_spectrum_in ซึ่งมีพารามิเตอร์สี่พารามิเตอร์ด้วยกันคือ

- ตัวแรกใช้กำหนดว่าจะแสดงภาพออกมาในลักษณะใด มีค่าได้ดังนี้คือ
  - shade จะแสดงภาพเป็นจุดสี่เหลี่ยมสีต่างๆ
  - gray จะแสดงภาพโดยใช้ความหนาแน่นของจุด
  - block แสดงภาพโดยใช้จุดสี่เหลี่ยมขนาดต่างๆ

- ตัวที่สองเป็นเมทริกซ์ที่ต้องการนำมาแสดงผล

- สองตัวสุดท้ายเป็นตำแหน่งบนจอภาพที่ต้องการให้นำภาพไปแสดงออก

อีกสองโพรซิเยอร์ที่ควรสนใจคือโพรซิเยอร์ reset_max_peak และ ฟังชัน max_peak_of_matrix ซึ่งใช้ในการคำนวณค่าสูงสุดของค่าสัมบูรณ์ของเมทริกซ์ที่เป็นพารามิเตอร์ของมัน โดยที่การคำนวณนี้จะทำเพียงครั้งเดียวเมื่อเรียกใช้ครั้งแรก หลังคำสั่ง reset_max_peak ซึ่งหมายว่าต้องการให้มีการคำนวณใหม่เมื่อเรียกใช้ฟังชัน max_peak_of_matrix

นอกจากนี้แล้วยังมีตัวแปรที่น่าสนใจอีกตัวหนึ่งได้แก่ตัวแปรที่ใช้ในการกำหนดระดับต่ำสุดที่จะนำไปใช้ในการเขียนภาพชื่อ graph_2d_base

โปรแกรมได้แสดงไว้ในหน้าถัดไป

```
unit graph_2d ;

(*                                      *)
(*    Filename  :  GRAPH_2D.PAS         *)
(*                                      *)

interface {section}

uses  complx , {unit of complex variable and function }
      vardef , {variable definition unit}
      graph  ,
      glue4  , {connect graphic command to simple command }
      ploter ;

type
    paint =
(shade,gray,contour,block,mix_shade,mix_gray,mix_block) ;


const
    gray_dot_color  : integer = 7 ;

    block_dot_color : integer = 7 ;

    graph_2d_base   : real    = 0.0 ;  { unit in % of
maximum peak }


procedure  ask_for_graph_2d_base ;

procedure  open_graph_window(xr1,yr1,xr2,yr2:integer) ;

procedure  draw_color_strip_at (x1,y1,x2,y2 ,color : integer
) ;

procedure  reset_max_peak ;

function   max_peak_of_matrix ( var u : matrix ) : real ;

procedure  display_power_spectrum_in ( mode  : paint     ;
                                       var u : matrix    ;
                                       {at}  x,y   : integer
) ;


implementation


var
    max_color  :  integer ;
    max_peak : real   ;
```

```
procedure  ask_for_graph_2d_base ;
var
     answer  : string  ;
     temp    : integer ;
begin
     { *** Ask for contour base level *** }
     writeln( ' Please enter graph base level .'     ) ;
     writeln( '  range in 0-100 %') ;
     writeln( '  default  : ',100.0*graph_2d_base:6:2,' %')
;
     write  ( '>> ' ) ; readln (answer) ;
     if answer <> null then
     begin
          val (answer,graph_2d_base,temp) ;
          graph_2d_base := graph_2d_base / 100.00 ;
     end ;
end ;


procedure open_graph_window(xr1,yr1,xr2,yr2:integer) ;
var    x1,x2,y1,y2  : integer ;
begin
     graphwindow(xr1,yr1,xr2,yr2) ;
     x1 := 0 ; y1 := 0 ;
     x2 := xr2-xr1 ;  y2 := yr2-yr1 ;
     draw(x1,y1,x2,y1,white);
     draw(x2,y1,x2,y2,white);
     draw(x2,y2,x1,y2,white);
     draw(x1,y2,x1,y1,white);
end;


procedure draw_color_strip_at (x1,y1,x2,y2 , color : integer
) ;
var
     i,j   ,
     thick ,
     width : integer ;
begin
     thick :=  (y2-y1) div  color ;
     y2 :=  color * thick + y1 ;
     width :=  (x2-x1) ;

     open_graph_window(x1,y1,x2,y2+2);

     for i :=  0 to color-1  do
     for j :=  0 to thick  do
          draw( 1,i*thick+j+1 , width-1,i*thick+j+1 , color-
i-1) ;
end ;
```

```
procedure reset_max_peak ;
begin
     max_peak := 0.0 ;
end ;



function  max_peak_of_matrix ( var u : matrix ) : real ;
var  i,j  :  integer ;
     temp ,
     max  :  real ;
begin
     if  max_peak = 0.0  then
       begin
          max := u^[0]^[0].re ;
          for i := 0 to max_data - 1  do
          for j := 0 to max_data - 1  do
          begin
               temp  :=  sqr(u^[j]^[i].re) +
sqr(u^[j]^[i].im)  ;
               if  max < temp  then  max :=  temp  ;
          end {loop} ;
          max_peak  :=  max ;
       end
     else
          max  :=  max_peak ;
    {end if} ;


    max_peak_of_matrix  := sqrt(max) ;

end;



             procedure dot_in_gray ( x,y , scale , color :
integer ) ;
             var
                   d,k       ,
                   i,j,n     : integer  ;
             begin
                  if color <> 0  then
                  begin
                       for i :=  0 to scale-1  do
                       for j :=  0 to scale-1  do
                       if random(max_color) < color  then
                            plot(x+i,y+j,gray_dot_color)
                       {end if} ;
                  end {if} ;
             end {dot_in_gray} ;
```

```
            procedure dot_in_block ( x,y , scale , color :
integer ) ;
            var
                  count     ,
                  shell     ,
                  cx,cy     ,
                  d,k       ,
                  i,j,n     : integer  ;
            begin
                 shell := trunc(sqrt( color ) ) ;
                 cx := (scale-shell) div 2 ;
                 cy := cx ;
                 count := color - shell*shell ;
                 for i :=  0 to shell-1 do
                 for j :=  0 to shell-1 do
                     plot( i+x+cx , j+y+cy ,
block_dot_color ) ;
                 i := 0 ;
                 repeat
                       if  count > 0  then
                           plot( i+x+cx , shell+y+cy ,
block_dot_color ) ;
                       count := count - 1 ;
                       if  count > 0  then
                           plot( shell+x+cx , i+y+cy ,
block_dot_color ) ;
                       count := count - 1 ;
                       i := i + 1 ;
                 until count <= 0 ;
            end {dot_in_gray} ;


            procedure dot_in_shade ( x,y , scale , color :
integer ) ;
            var  k : integer ;
            begin
                 if color <> 0 then
                    for k := 0 to scale-1 do
                        draw( x,y+k , x+scale-1,y+k, color)
;
                    {end loop}
                 {end if}
            end {dot_in_shade} ;
```

```
procedure  display_power_spectrum_in ( mode  : paint     ;
                                       var u : matrix    ;
                                       {at}  x,y   : integer
) ;
var   i,j,k ,
      scale ,
      color :  integer ;
      temp  ,
      max   :  real    ;
      base_level : real ;

begin

     {  AUTOMATIC SCALE  }

     scale := GetMaxY div max_data ;

     if  scale < 1  then  scale  := 1    ;

     { CALCULATE MAX_COLOR  }
     case  mode  of
          shade  :  begin
                         max_color := GetMaxColor+1 ;

                    {  DRAW REFERANCE SHADE STRIP  }

                         draw_color_strip_at
                         ( x+(scale*(max_data))+10,
                           y+scale*(max_data)-max_color*16,
                           x+(scale*(max_data))+39,
                           y+scale*(max_data)+1 ,
                           max_color ) ;
                    end ;

      gray,block  : max_color :=  scale*scale ;

     end {case} ;


     max := 1.01 * max_peak_of_matrix( u ) ;

     base_level := graph_2d_base * max ;

     open_graph_window( x,y, x+(scale*(max_data))+1 ,
y+scale*(max_data)+1 ) ;

     for j := max_data-1 downto 0 do
     for i := 0 to max_data-1  do
     begin
          temp   :=  sqrt( sqr(u^[i]^[j].re) +
sqr(u^[i]^[j].im) ) ;
          temp   :=  temp - base_level ;
```

```
          if  temp < 0.0  then temp := 0.0 ;

          color  :=  trunc ( temp *  max_color / max )    ;
          case mode of
              shade : dot_in_shade( i*scale+1,(max_data-j-
1)*scale+1 ,
                                    scale , color);
               gray : dot_in_gray ( i*scale+1,(max_data-j-
1)*scale+1 ,
                                    scale , color);
              block : dot_in_block( i*scale+1,(max_data-j-
1)*scale+1 ,
                                    scale , color);
          end {case} ;
     end {loop} ;

     if  adeptor = plotter  then  flush_plotter_buffer ;
end ;




begin {init section }

      reset_max_peak
end..
```

ภาคผนวก ค

ยูนิต massageg

คำสั่งที่สร้างขึ้นในยูนิตนี้ จะเป็นคำสั่งที่ใช้ในการแสดงข้อความในขณะที่มีการแสดงภาพบนจอภาพ

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจดังนี้

1. โพรซิเยอร์ out_text_in มีพารามิเตอร์เป็นตัวเลขจำนวนเต็มสี่ตัว และข้อความหนึ่งตัว ใช้ในการนำข้อความที่กำหนดไปแสดงออกทางจอภาพโดยที่ข้อความนี้จะมีขนาดพอดีกับบริเวณที่กำหนดโดยพารามิเตอร์สี่ตัวแรก

2. โพรซิเยอร์ display_massage_on มีพารามิเตอร์เป็นตัวเลขจำนวนเต็มสี่ตัว และข้อความซึ่งเป็นชื่อแฟ้มข้อมูลหนึ่งตัว ใช้ในการนำชื่อแฟ้มข้อมูลที่กำหนด และข้อความว่า

NMR

laboratory

Chulalongkorn

University

ไปแสดงออกทางจอภาพโดยที่ข้อความนี้จะมีขนาดพอดีกับบริเวณที่กำหนดโดยพารามิเตอร์สี่ตัวแรก

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไปแล้ว

```
unit massageg ;

interface

uses cell,graph,glue4 ;

procedure display_massage_on ( x1,y1 ,x2,y2 : integer ;
filename : string ) ;

procedure out_text_in ( x1,y1,x2,y2 : integer ; massage :
string ) ;


implementation


var     text_info : textsettingstype ;
        greee : integer ;


procedure out_text_in ( x1,y1,x2,y2 : integer ; massage :
string ) ;

var  h,w   : word ;

begin
     SetUserCharSize(1,1,1,1);
     SetTextJustify ( LeftText , TopText ) ;

     h :=  TextHeight (massage) ;
     w :=  TextWidth  (massage) ;

     SetUserCharSize( abs(x2-x1)+1 ,w, abs(y2-y1)+1 ,h);

     OutTextXY ( x1,y1 , massage ) ;
end ;



procedure OutText_nmr_in ( x1,y1,x2,y2 : integer ) ;

var  h,w      ,
     size     : integer ;
     size_OK : boolean ;

begin
     size := 10 ;
     repeat
           SetTextStyle( DefaultFont , HorizDir, size ) ;
           h :=  TextHeight ('NMR') ;
           w :=  TextWidth  ('NMR') ;
           size_OK := ( h < abs(y1-y2)+1 ) and ( w < abs(x2-
x1)+1 ) ;
```

```
            dec (size) ;
      until size_OK or (size<1) ;

      if size_ok then
      begin
            SetTextJustify ( CenterText , CenterText ) ;
            OutTextXY ( (x1+x2)div 2 ,(y2+y1)div 2, 'NMR') ;
      end {if} ;
end ;



procedure display_massage_on ( x1,y1 ,x2,y2 : integer ;
filename : string ) ;

const  m1 = 'Image from FILE' ;

var w1,w2,c : integer ;


begin
      if adeptor = plotter then exit ;

      SetColor (15) ;

      w1 := round (y1+0.00*(y2-y1)) ;
      w2 := round (y1+0.05*(y2-y1)) ;

      SetTextStyle( TriplexFont , HorizDir, UserCharSize ) ;

      Out_text_in ( x1,w1 , x2,w2 , m1) ;

      w1 := round (y1+0.06*(y2-y1)) ;
      w2 := round (y1+0.13*(y2-y1)) ;

      Out_Text_in ( x1,w1 , x2,w2 , filename) ;

      SetColor (2) ;
      w1 := round (y1+0.50*(y2-y1)) ;
      w2 := round (y1+0.65*(y2-y1)) ;
      OutText_nmr_in ( x1,w1,x2,w2 ) ;


      SetTextStyle( SansSerifFont , HorizDir, UserCharSize )
;

      w1 := round (y1+0.65*(y2-y1)) ;
      w2 := round (y1+0.72*(y2-y1)) ;

      Out_Text_in ( x1,w1 ,x2,w2, 'Laboratory') ;

      SetColor (5) ;
      SetTextStyle( GothicFont , HorizDir, 3 ) ;
```

```
      w1 := round (y1+0.85*(y2-y1)) ;
      w2 := round (y1+0.91*(y2-y1)) ;

      Out_Text_in( x1,w1 , x2,w2 , 'Chulalongkorn') ;


      w1 := round (y1+0.91*(y2-y1)) ;
      w2 := round (y1+0.97*(y2-y1)) ;

      Out_Text_in( x1+10,w1 , x2-10,w2 , 'University') ;

end ;


begin
end .
```

ภาคผนวก ต

ยูนิต menu

ในการทำงานในลักษณะที่มีการใช้งานเมนูเพื่อให้ผู้ใช้เลือกใช้งานนั้น ได้สร้างคำสั่งขึ้นในยูนิตนี้ เพื่อจะเป็นคำสั่งที่ใช้ในการควบคุมและอำนวยความสะดวกในการทำงานเกี่ยวกับเมนูทั้งหลาย

ข้อมูลและคำสั่งที่สร้างในยูนิต

ในยูนิตนี้ได้มีการกำหนดประเภทของตัวแปรขึ้นมาใหม่ประเภทหนึ่ง ซึ่งหากจะใช้งานคำสั่งเกี่ยวกับเมนูที่สร้างขึ้นแล้ว ก็จำเป็นต้องทราบเกี่ยวกับตัวแปรประเภทนี้พอสมควร ประเภทของตัวแปรที่กำหนดขึ้นมานี้คือ menu_type ซึ่งจะใช้ในการเก็บข้อมูลเกี่ยวกับเมนูทั้งหลาย และโดยที่เมนูอาจมีจำนวนตัวเลือกได้ไม่แน่นอน ดังนั้น จึงได้กำหนดให้ตัวแปรประเภทนี้เป็นตัวแปรพลวัต (dynamic variable) ซึ่งสามารถโปรแกรมให้ขนาดของตัวแปรมีขนาดแปรไปได้ตามต้องการ

ในยูนิตนี้มีคำสั่งที่เกี่ยวข้องกับการใช้งานเมนูดังนี้

1. โพรซีเยอร์ new_menu มีพารามิเตอร์หนึ่งตัวเป็นประเภท menu_type โดยโพรซีเยอร์นี้จะทำหน้าที่เตรียมตัวแปรที่ส่งผ่านเข้ามาให้พร้อมที่จะนำไปใช้งาน แล้วจึงส่งกลับ

2. โพรซีเยอร์ add_menu มีพารามิเตอร์สามตัว โพรซีเยอร์นี้จะใช้สำหรับเพิ่มตัวเลือกของเมนูตามพารามิเตอร์ชนิดข้อความตัวที่สาม เข้าไปในตัวแปรที่เป็นพารามิเตอร์ตัวแรกซึ่งเป็นประเภท menu_type โดยที่ตัวเลือกนี้จะกำหนดคีย์ที่ใช้เลือกโดย

พารามิเตอร์ตัวที่สอง

3.โพรซิเยอร์ menu_normal_color, โพรซิเยอร์ menu_inverse_color, โพรซิเยอร์ menu_highlight_color ทั้งสามโพรซิเยอร์นี้จะใช้ในการกำหนดสีที่จะใช้ในขณะนำเมนูออกแสดง โดยจะใช้กำหนดสีของ เมนูปรกติ, แถบสว่างที่ใช้ชี้เลือก, และตัวอักษรที่ใช้เลือก ตามลำดับ

4. โพรซิเยอร์ select_menu มีพารามิเตอร์ห้าตัวด้วยกัน โพรซิเยอร์นี้จะใช้สำหรับแสดงตัวเลือกของเมนูตามตัวแปรที่เป็นพารามิเตอร์ตัวแรกซึ่งเป็นประเภท menu_type ออกมาบนจอภาพ ณ ตำแหน่งที่กำหนดโดยพารามิเตอร์ตัวที่สี่และห้า เมื่อผู้ใช้เลือกหัวข้อใดหัวข้อหนึ่งก็จะส่งหมายเลขของหัวข้อที่เลือกผ่านกลับทางพารามิเตอร์ตัวที่สอง พร้อมกับส่งคีย์ที่กดเพื่อเลือกผ่านกลับทางพารามิเตอร์ตัวที่สาม

5. โพรซิเยอร์ select_menu_bar มีพารามิเตอร์ห้าตัวด้วยกัน โพรซิเยอร์นี้จะใช้สำหรับแสดงตัวเลือกของเมนูตามตัวแปรที่เป็นพารามิเตอร์ตัวแรกซึ่งเป็นประเภท menu_type ออกมาบนจอภาพ ณ ตำแหน่งที่กำหนดโดยพารามิเตอร์ตัวที่สี่และห้า โดยจะต่างจากโพรซิเยอร์ select_menu ตรงที่การแสดงออกมานี้จะวางเมนูเรียงกันไปตามแนวนอน แทนที่จะเป็นแนวตั้ง และเมื่อผู้ใช้เลือกหัวข้อใดหัวข้อหนึ่งก็จะส่งหมายเลขของหัวข้อที่เลือกผ่านกลับทางพารามิเตอร์ตัวที่สอง พร้อมกับส่งคีย์ที่กดเพื่อเลือกผ่านกลับทางพารามิเตอร์ตัวที่สาม

ในการนำคำสั่งเหล่านี้ไปใช้ช่วยในการเขียนโปรแกรม จะเริ่มที่ผู้เขียนจะต้องกำหนดตัวแปรขึ้นมาโดยให้มีประเภทเป็น menu_type แล้วจึงสั่ง new_menu จากนั้นจึงเพิ่มรายการเข้าไปทีละรายการโดยคำสั่ง add_menu เมื่อพอเพียงแล้วจึงนำตัวแปรที่ได้ไปใช้แสดงให้ผู้ใช้เลือกโดยคำสั่ง select_menu

โปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไป

```
unit menu ;

(**************************************)
(**                                  **)
(**     Filename   :     MENU.PAS     **)
(**                                  **)
(**************************************)

(***
      The variable typed "menu_type" must be define with

       CONST  <iden> : menu_type = nil ;
***)


interface


uses  crt,cell,
      windsys ,mouse ;  { This unit is not neccesary }

type
       menu_string =  string[80]  ;
       menu_type   = ^menu_status ;
       menu_text   = ^menu_infor ;

       menu_infor  =  record
                           text    : menu_string ;
                           sel     : char        ;  { char
that highlight }
                           col     : byte        ;
                           ptr     : menu_text   ;
                      end ;
       menu_status =  record
                           memo             : byte ;
                           normal_color     : byte ;
                           inverse_color    : byte ;
                           highlight_color  : byte ;
                           menu_massage     : menu_text ;
                      end ;

var  last_menu_scan_key : char ;

procedure new_menu ( var menu : menu_type ) ;

procedure add_menu ( menu     : menu_type ;
                     selector : char ;
                     massage  : menu_string ) ;

procedure menu_normal_color    ( menu : menu_type ; fore ,
back : byte ) ;
```

```
procedure menu_inverse_color   ( menu : menu_type ; fore ,
back : byte ) ;

procedure menu_highlight_color ( menu : menu_type ; fore ,
back : byte ) ;

procedure select_menu ( menu : menu_type ;
                        var select_item : byte   ;
                        var select_key  : char   ;
                  {at}  x,y  :  byte ) ;

procedure select_menu_bar ( menu : menu_type ;
                            var select_item : byte     ;
                            var select_key  : char     ;

                      {at}  x,y  :  byte ) ;


implementation


var
      attr_save : byte ;


function blank ( n : byte ) : string ;
var x : string ;
    a : byte absolute x ;
begin
     FillChar ( x , n+1 , #32 ) ;
     a := n ;
     blank := x ;
end ;

procedure menu_normal_color ( menu : menu_type ; fore , back
: byte ) ;
begin
     back  :=  back and 7 ;
     menu^.normal_color    := back*16 + fore ;
     menu^.inverse_color   := back*16 + fore ;
     menu^.highlight_color := back*16 + fore ;
end ;

procedure menu_inverse_color ( menu : menu_type ; fore ,
back : byte ) ;
begin
     menu^.inverse_color := back*16 + fore ;
end ;

procedure menu_highlight_color ( menu : menu_type ; fore ,
back : byte ) ;
begin
     menu^.highlight_color := back*16 + fore ;
end ;
```

```
procedure save_attr ;
begin
     attr_save := Crt.TextAttr ;
end ;

procedure restore_attr  ;
begin
     Crt.TextAttr := attr_save ;
end ;

procedure normal ( menu : menu_type ) ;
begin
     Crt.TextAttr :=  menu^.normal_color ;
end ;

procedure inverse ( menu : menu_type ) ;
begin
     Crt.TextAttr :=  menu^.inverse_color ;
end ;

procedure highlight ( menu : menu_type ) ;
begin
     Crt.TextAttr :=  menu^.highlight_color ;
end ;


procedure new_menu ( var menu : menu_type ) ;
var
   now,next  : menu_text ;
begin
     if  menu = nil  then
     begin
          new (menu) ;
          menu^.menu_massage := nil ;
          menu_normal_color ( menu , Lo(TextAttr) ,
Hi(TextAttr) ) ;
     end
     else
     begin
          next :=  menu^.menu_massage ;
          menu^.menu_massage := nil ;
          if  next <> nil  then
          repeat
                now  := next ;
                next := next^.ptr ;
                dispose(now) ;
          until  next = nil ;
     end {if} ;

end {new_menu};
```

```
procedure add_menu ( menu     : menu_type ;
                     selector : char ;
                     massage  : menu_string ) ;
var
   mem_need  : longint ;
   now,last  : menu_text ;
begin
     mem_need := SizeOf (menu_text)  ;
     with  menu^  do

     if menu_massage = nil then

       if maxavail > mem_need then
         begin
           new( menu_massage ) ;
           menu_massage^.text := massage ;
           menu_massage^.sel  := selector ;
           menu_massage^.col  := pos( selector , massage ) ;
           menu_massage^.ptr  := nil     ;
         end
       else
           out_of_memory
       {end if}

     else

       if maxavail > mem_need then
         begin
           now  := menu_massage ;
           while  now^.ptr <> nil  do  now := now^.ptr ;

           new( now^.ptr ) ;
           now := now^.ptr ;
           now^.text := massage ;
           now^.sel  := selector ;
           now^.col  := pos( selector , massage ) ;
           now^.ptr  := nil     ;
         end
       else
           out_of_memory ;
       {end if} ;

     {end if} ;
     {end with}

end {add_menu} ;

type  menu_data  =  array [1..25] of
                    record
                          menu_line : menu_string ;
                          sel_col   : byte ;
                          selector  : char ;
                    end ;
```

```
procedure dump_menu ( menu : menu_type ;
                {to} var information  : menu_data ;
                     var data_num     : byte ;
                     var width        : byte  ) ;
var
   now       : menu_text ;
   again     : boolean ;
   i,n       : byte ;
begin
     n    := 0 ;
     now  := menu^.menu_massage ;
     again := true ;

     repeat
          inc (n) ;
          with  information[n]  do
          begin
               menu_line := now^.text ;
               sel_col   := now^.col  ;
               selector  := now^.sel  ;
          end {with} ;
          if  now^.ptr = nil  then again := false ;
          now := now^.ptr ;
     until  not again ;

     data_num := n ;

     width := 0 ;
     for i := 1 to n do
       with  information[i]  do
          if width < length(menu_line) then  width :=
length(menu_line) ;

     for i := 1 to n do
        with  information[i]  do
          menu_line := menu_line + blank (width-
length(menu_line)) ;

end {dump_menu} ;


procedure check_for_selector_key ( information : menu_data ;
data_num : byte ;
                                   var inkey : char ;
                                   var select_item : byte )
;
var  k : byte ;
     again : boolean ;
begin
     k := 1 ;
     select_item := 0 ;
```

```
     repeat
           with information[k] do
           begin
                again := true ;
                if ( upper_case (inkey) = upper_case
(selector) )
                and  ( sel_col <> 0 ) then
                begin
                     again := false ;
                     inkey := return ;
                     select_item := k  ;
                end {if};
                inc (k) ;
           end {with} ;
     until  (not again) or (k>data_num) ;

   { this procedure
     -return select_item = 0 if inkey isn't selector.
     -return select_item = n if inkey is selector of 'n'
menu.
   }
end {check_for_selector_key} ;



procedure select_menu ( menu : menu_type ;
                        var select_item : byte     ;
                        var select_key  : char     ;
                  {at}  x,y  :  byte ) ;
var
   information       : menu_data ;
   item              ,
   i,j,n             : byte ;
   scan,inkey        : char ;
   width             : byte ;

   not_again         ,
   mouse_active      :  boolean ;

   mouse_pos_x       ,
   mouse_pos_y       :  integer ;


           procedure  write_menu ( n : byte ) ;
           begin
                with  information[n]  do
                begin
                     normal (menu) ;
                     gotoxy( x , y+n-1 ) ;
                     OutString ( menu_line ) ;
```

```
                    if  sel_col <> 0  then
                    begin
                         highlight (menu);
                         gotoxy( sel_col+x-1 , y+n-1 ) ;
                         OutChar ( selector ) ;
                    end {if} ;
               end {with} ;
          end { write_menu } ;


     procedure  move_pointer_from_j_to_i ;
          begin
               write_menu (j) ;

               inverse (menu) ;
               gotoxy( x , y+i-1 ) ;
               OutString( information[i].menu_line ) ;
               j := i ;  { j is old pointer position }
          end  {move_pointer_from_j_to_i} ;


          procedure  convert_to_line_column ( var
mouse_pos_x , mouse_pos_y : integer ) ;
          begin
               mouse_pos_x := (mouse_pos_x div 8) + 1 ;
               mouse_pos_x := mouse_pos_x - Lo(WindMin) -
x + 1 ;
               mouse_pos_y := (mouse_pos_y div 8) + 1 ;
               mouse_pos_y := mouse_pos_y - Hi(WindMin) -
y + 1 ;
          end ;

          function   pos_x_in_range : boolean ;
          begin
               pos_x_in_range :=  (mouse_pos_x > 0)    and
                                  (mouse_pos_x <= width) ;
          end ;

          function   pos_y_in_range : boolean ;
          begin
               pos_y_in_range :=  (mouse_pos_y > 0) and
                                  (mouse_pos_y <= n) ;
          end ;

const
   numeric_key_scan : set of char =
[#$4F,#$50..#$52,#$4B..#$4D,#$47..#$49] ;

begin
     if menu = nil then exit ;


     save_attr ;
     hide_cursor ;
```

```
     dump_menu ( menu , information , n ,width ) ;


     normal (menu) ;
     for i := 1 to n do write_menu (i) ;

   { i store where pointer point to }{ j store old value of
i }

     if menu^.memo in [1..n]  then  i := menu^.memo  else  i
:= 1 ;

     j := i ;

     move_pointer_from_j_to_i ;

     not_again := false ;

     repeat
           clear_button_press ;
           show_icon ;
           wait_mouse_or_key ;
           hide_icon ;

           if  KeyPressed  then
           begin
                scan_key_to ( scan , inkey ) ;

                { check for directional key }
                if scan in numeric_key_scan then inkey :=
null  ;

                case  scan  of
                    UpArrow  : i := i - 1 ;
                   DownArrow : i := i + 1 ;
                end {case} ;

                if  i > n then i := 1 ;
                if  i < 1 then i := n ;

                check_for_selector_key ( information,n ,
inkey , item ) ;
                if  item <> 0  then  i := item ;

                move_pointer_from_j_to_i ;

                not_again :=
                      (inkey in
[return,ESC,space,'0'..'9','a'..'z','A'..'Z'])
                      or (scan=LeftArrow)
                      or (scan=RightArrow) ;
           end {if} ;
```

```
            if  pressing_mouse_button (left) then
            begin
                 get_press_pos_of_button (left
,mouse_pos_x,mouse_pos_y ) ;
                 convert_to_line_column ( mouse_pos_x ,
mouse_pos_y ) ;

                 if  pos_x_in_range  and  (mouse_pos_y = i)
then
                   begin
                      inkey := return ;
                      not_again := true ;
                   end
                 else
                   begin
                      show_icon ;
                      repeat
                           if  pos_x_in_range and
pos_y_in_range  and
                               (mouse_pos_y <> i)     then
                           begin
                                i := mouse_pos_y ;
                                hide_icon ;
                                move_pointer_from_j_to_i ;
                                show_icon ;
                           end {if} ;

                           mouse_active  :=
pressing_mouse_button (left) ;
                           get_press_pos_of_button (left ,

mouse_pos_x,mouse_pos_y );
                           convert_to_line_column (
mouse_pos_x , mouse_pos_y );

                      until not mouse_active ;
                      not_again := false ;
                      hide_icon ;
                   end {if}
                 {end if} ;
        end {if} ;

            if pressing_mouse_button (middle)  then
            begin
                 inkey := Esc ;
                 not_again := true ;
            end ;


     until  not_again ;
```

```
      if (scan=LeftArrow) or (scan=RightArrow) then
        last_menu_scan_key := scan
      else
        last_menu_scan_key := null
      {end if} ;

      menu^.memo := i ;

      select_item := i ;
      select_key  := inkey ;

      show_cursor ;
      restore_attr ;

end {select_menu} ;


procedure select_menu_bar ( menu : menu_type ;
                            var select_item : byte     ;
                            var select_key  : char     ;

                        {at}  x,y  :  byte ) ;
var
   information : menu_data ;
   width      ,
   icon_width ,
   item       ,

   i,j,n,k    : byte ;
   again      : boolean ;
   scan       ,
   inkey      : char ;

   not_again         ,
   mouse_active      :  boolean ;

   mouse_pos_x       ,
   mouse_pos_y       :  integer ;

            procedure  write_menu ( n : byte ) ;
            begin
                 with  information[n]  do
                 begin
                      normal (menu) ;
                      gotoxy( x+(n-1)*width , y ) ;
                      OutString( copy (menu_line,1,width)  )
;
```

```
                    if  (sel_col <> 0) and (sel_col <
width)  then
                    begin
                         highlight (menu);
                         gotoxy( sel_col+x-1+(n-1)*width ,
y ) ;
                         OutChar ( selector ) ;
                    end {if} ;
               end {with} ;
          end { write_menu } ;


     procedure  move_pointer_from_j_to_i ;
          begin
               write_menu (j) ;

               inverse (menu) ;
               gotoxy( x+(i-1)*width , y ) ;
               OutString( information[i].menu_line ) ;
               j := i ;  { j is old pointer position }
          end  {move_pointer_from_j_to_i} ;


          procedure  convert_to_line_column ( var
mouse_pos_x , mouse_pos_y : integer ) ;
          begin
               mouse_pos_x := (mouse_pos_x div 8) + 1 ;
               mouse_pos_x := mouse_pos_x - Lo(WindMin) -
x + 1 ;
               mouse_pos_y := (mouse_pos_y div 8) + 1 ;
               mouse_pos_y := mouse_pos_y - Hi(WindMin) -
y + 1 ;
          end ;

          function   pos_x_and_y_in_range : boolean ;
          begin
               pos_x_and_y_in_range :=  ( mouse_pos_y = 1
) and
                  ( mouse_pos_x > Lo(WindMin)   ) and
                  ( mouse_pos_x < Lo(WindMax)+2 ) ;

          end ;

          function  item_mouse_on : integer ;
          begin
               item_mouse_on := (mouse_pos_x div width) +
1  ;
          end ;


const
   numeric_key_scan : set of char =
[#$4F,#$50..#$52,#$4B..#$4D,#$47..#$49] ;
```

```
begin
     if menu = nil then exit ;

     save_attr ;
     hide_cursor ;
     normal (menu);

     dump_menu ( menu , information , n , icon_width ) ;


     gotoxy (1,y) ;
     OutString( blank (Lo(WindMax)-Lo(WindMin)+1) )  ;

     width := ( Lo(WindMax) - Lo(WindMin) + 1 ) div n  ;


     for i := 1 to n do  write_menu (i) ;

     if menu^.memo in [1..n]  then  i := menu^.memo  else  i
:= 1 ;

     j := i ;

     if (last_menu_scan_key=LeftArrow) or
(last_menu_scan_key=RightArrow)
     then begin
          case last_menu_scan_key of
               LeftArrow : i := i-1 ;
               RightArrow : i := i+1 ;
          end {case} ;
     end {if} ;

     if  i > n then i := 1 ;
     if  i < 1 then i := n ;

     move_pointer_from_j_to_i ;

     not_again := false ;

     repeat
           clear_button_press ;
           show_icon ;
           wait_mouse_or_key ;
           hide_icon ;

           if  KeyPressed  then
           begin
                scan_key_to ( scan , inkey ) ;

                { check for directional key }
                if scan in numeric_key_scan then inkey :=
null  ;
```

```
                case  scan  of
                   LeftArrow : i := i - 1 ;
                  RightArrow : i := i + 1 ;
                   DownArrow : inkey := return ;
                end {case} ;


                if  i > n then i := 1 ;
                if  i < 1 then i := n ;

                check_for_selector_key ( information,n ,
inkey , item ) ;
                if  item <> 0  then  i := item ;

                move_pointer_from_j_to_i ;

                not_again :=
                      (inkey in
[return,ESC,space,'0'..'9','a'..'z','A'..'Z']);

           end {if} ;


           if pressing_mouse_button (left) then
           begin
                get_press_pos_of_button (left
,mouse_pos_x,mouse_pos_y ) ;
                convert_to_line_column ( mouse_pos_x ,
mouse_pos_y ) ;

                if (item_mouse_on = i) and  (mouse_pos_y =
1)  then
                  begin
                     inkey := return ;
                     not_again := true ;
                  end
                else
                  begin
                     show_icon ;
                     repeat
                          if  pos_x_and_y_in_range and
                              (item_mouse_on <> i) and
                              (item_mouse_on <= n)   then
                          begin
                               i := item_mouse_on ;
                               hide_icon ;
                               move_pointer_from_j_to_i ;
                               show_icon ;
                          end {if} ;

                          mouse_active  :=
pressing_mouse_button (left) ;
```

```
                                get_press_pos_of_button (left ,
mouse_pos_x,mouse_pos_y );
                                convert_to_line_column (
mouse_pos_x , mouse_pos_y );

                        until not mouse_active ;
                        not_again := false ;
                        hide_icon ;
                    end {if}
                {end if} ;
        end {if} ;

            if pressing mouse button (middle) then
            begin
                inkey := Esc ;
                not_again := true ;
            end ;

    until  not_again ;


    menu^.memo := i ;

    select_item := i ;
    select_key  := inkey ;

    last_menu_scan_key := null ;

    show_cursor ;
    restore_attr;

end {select_menu_bar} ;



begin {init section }

end .
```

ภาคผนวก ก

ยูนิต mouse

ในยูนิตนี้จะเป็นคำสั่งที่สร้างขึ้นเพื่อใช้ในงานทั่วการควบคุมและเรียกใช้งานเมาส์(mouse)

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจอยู่มากมายดังต่อไปนี้

1. โพรซิเยอร์ reset_mouse ไม่มีพารามิเตอร์ ใช้ในการเตรียมเมาส์ก่อนจะใช้งาน

2. โพรซิเยอร์ show_icon ไม่มีพารามิเตอร์ จะทำให้ตัวชี้ของเมาส์ปรากฏบนจอภาพ

3. โพรซิเยอร์ hide_icon ไม่มีพารามิเตอร์ จะทำให้ตัวชี้ของเมาส์ไม่ปรากฏบนจอภาพ

4. โพรซิเยอร์ show_graph_icon ไม่มีพารามิเตอร์ จะทำให้ตัวชี้ของเมาส์ปรากฏบนจอภาพแบบกราฟิก

5. โพรซิเยอร์ hide_graph_icon ไม่มีพารามิเตอร์ จะทำให้ตัวชี้ของเมาส์ไม่ปรากฏบนจอภาพจอภาพแบบกราฟิก

6. ฟังชัน mouse_button_presses มีพารามิเตอร์เป็นเลขหมายของปุ่มบนเมาส์ที่ต้องการทราบว่ามีการกดหรือไม่ โดยจะส่งจำนวนครั้งที่มีการกดทั้งหมดก่อนที่จะมีการเรียกฟังชันนี้ กลับมาทางชื่อของฟังชัน

7. ฟังชัน pressing_mouse_button มีพารามิเตอร์เป็นเลขหมายของปุ่มบนเมาส์ที่ต้องการทราบว่ามีการกดหรือไม่ โดยจะส่งผลว่ามีการกดหรือไม่ ในขณะที่เรียกฟังชันนี้กลับผ่านทางชื่อของฟังชัน

8. ฟังชัน mouse_button_pressing ไม่มีพารามิเตอร์ จะส่งผลว่ามีการกดปุ่มใดๆบนเมาส์หรือไม่ในขณะที่เรียกฟังชันนี้ กลับผ่านทางชื่อของฟังชัน

9. โพรซิเยอร์ get_press_pos_of_button ใช้อ่านว่ามีการกดปุ่ม (หมายเลขตามพารามิเตอร์แรก) บนเมาส์ที่ตำแหน่งใด โดยที่จะส่งค่ากลับผ่านพารามิเตอร์ตัวที่สองและสาม

10. โพรซิเยอร์ set_mouse_bound เป็นโพรซิเยอร์ที่ใช้กำหนดขอบเขตของตัวชี้ของเมาส์บนจอภาพ ตามที่ระบุมาในพารามิเตอร์ทั้งสี่ตัว

11. โพรซิเยอร์ define_text_cursor เป็นโพรซิเยอร์ที่ใช้กำหนดลักษณะและสีของตัวชี้ของเมาส์ที่ปรากฏบนจอภาพ

12. โพรซิเยอร์ wait_mouse_or_key จะทำให้โปรแกรมหยุดรอจนกว่าจะมีการกดปุ่มบนเมาส์ หรือบนแป้นพิมพ์

13. โพรซิเยอร์ set_Mickey_ratio มีพารามิเตอร์สองตัวซึ่งจะใช้กำหนดความเร็วในการเคลื่อนที่ของตัวชี้ของเมาส์ในแนวแกนนอน และในแนวแกนตั้งตามลำดับ

14. โพรซิเยอร์ move_icon ไม่มีพารามิเตอร์ เนื่องจากในระหว่างการทำงานของโปรแกรมนั้น ตัวชี้ของเมาส์ที่อยู่บนจอภาพนี้จะต้องเคลื่อนที่ตามการเคลื่อนที่ของเมาส์ไปตลอดเวลา จึงได้สร้างโพรซิเยอร์นี้ขึ้นเพื่อนำไปแทรกไว้ในส่วนต่างๆของโปรแกรม ซึ่งเมื่อโปรแกรมวิ่งมาพบคำสั่งนี้ก็จะเลื่อนตัวชี้ของเมาส์ไปตามการเคลื่อนที่ของเมาส์ให้

15. โพรซิเยอร์ new_icon มีพารามิเตอร์ซึ่งเป็นตัวแปรที่ใช้เก็บรูปของตัวชี้ของเมาส์ในแบบกราฟฟิก โพซิเยอร์นี้จะทำการเตรียมตัวแปรให้พร้อมที่จะใช้งานใหม่ได้

16. โพรซิเยอร์ read_icon_from_file จะอ่านรูปร่างของตัวชี้ของเมาส์ในแบบกราฟฟิกขึ้นมาจากแฟ้มข้อมูลตามพารามิเตอร์ตัวแรก แล้วบันทึกไว้บนตัวแปรตามพารามิเตอร์ตัวที่สอง

17. โพรซิเยอร์ select_graph_icon จะใช้เลือกให้ใช้รูปของรูปร่างของตัวชี้ของเมาส์ในแบบกราฟฟิกตามพารามิเตอร์ในการชี้บนจอภาพ

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไป

```
{$Define user_icon}
unit  mouse ;


(***
      The variable typed "icon_type" must be define with

       CONST  <iden> : icon_type = nil ;
***)


interface

uses  dos,crt,graph ;


const   left   = 0 ;
        middle = 2 ;
        right  = 1 ;


type
       icon_record =  record
                        size   :  word    ;
                        x1,y1  :  integer ;
                        x2,y2  :  integer ;
                        shape1 :  ^byte   ;
                        shape2 :  ^byte   ;
                      end ;

       icon_type   =  ^icon_record ;

procedure reset_mouse ;

procedure show_icon ;

procedure hide_icon ;

procedure show_graph_icon ;

procedure hide_graph_icon ;

procedure get_mouse_pos ( var x,y : integer ) ;

procedure set_mouse_pos ( x,y : integer ) ;


function  mouse_button_presses  ( button_no : byte ) :
integer ;

function  pressing_mouse_button ( button_no : byte ) :
boolean ;
```

```
function  mouse_button_pressing  : boolean ;

procedure clear_button_press ;

procedure set_button_wait_time ( time : integer ) ;

procedure get_press_pos_of_button ( button_no : word ; var
x,y : integer );

procedure set_mouse_bound ( x1,y1,x2,y2  : integer ) ;

procedure read_mouse_motion ( var x,y : integer ) ;

procedure define_text_cursor ( screen_mask , cursor_mask :
word ) ;

procedure wait_mouse_or_key  ;

procedure set_Mickey_ratio ( x,y : word ) ;

procedure set_threshold ( x : word ) ;

procedure move_icon ;

procedure  new_icon ( var icon : icon_type ) ;

procedure  read_icon_from_file ( fname : string ; var icon :
icon_type ) ;

procedure  select_graph_icon ( var icon : icon_type ) ;


const    mouse_is_installed      :  boolean  =  false ;

implementation

const
      video_service = $10 ;
      mouse_int     =  $33  ;

      x_pos_of_mouse_presses : array [0..2] of integer =
(0,0,0)  ;
      y_pos_of_mouse_presses : array [0..2] of integer =
(0,0,0)  ;

      wait_time_for_next_presses : integer = 10 {millisec} ;
```

```
      display_text_icon   :  boolean  =  false  ;
      display_graph_icon  :  boolean  =  false  ;

      text_cursor_screen_mask  :  word  =  0  ;
      text_cursor_cursor_mask  :  word  =  0  ;

      old_x : integer = 0 ;
      old_y : integer = 0 ;


procedure reset_mouse ;
var
   regs  :  Registers ;
begin
     with  regs  do
     begin
          AX := 0 ;
       intr ( mouse_int , regs ) ;
       if  ax <> 0   then
             mouse_is_installed  :=  true
          else
             mouse_is_installed  :=  false
          {end if} ;
     end {with} ;
end ;


procedure  get_mouse_coordinate (var x,y : integer ) ;
var
   regs  :  Registers ;
begin
     regs.AX := 3 ;
     intr ( mouse_int , regs ) ;
     x := regs.CX div 2 ;
     y := regs.DX ;
end ;



procedure  set_cursor_to ( col,row : word ) ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $02 ;
          DL := col ;
          DH := row ;
          intr ( video_service , regs ) ;
     end {with} ;
end ;
```

```
function display_page : byte ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $0F ;
          intr ( video_service , regs ) ;
          display_page := BH ;
     end {with} ;
end ;


procedure  GetChar ( var char,attr : byte ) ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $08 ;
          BH := display_page ;
          intr ( video_service , regs ) ;
          char := AL ;
          attr := AH ;
     end {with} ;
end ;


procedure  PutChar ( char,attr : byte ) ;
var  regs : Registers ;
begin
     with  regs  do
     begin
          AH := $09 ;
          AL := char ;
          BH := display_page ;
          BL := attr ;
          CX := 1 ;      { only one character sended to
screen }
          intr ( video_service , regs ) ;
     end {with} ;
end ;



const   old_char : byte = 00 ;
        old_attr : byte = 00 ;
```

```
procedure  mask_icon ( var char,attr : byte ) ;
var  x : word ;
begin
     x  :=  (attr * $100) + char   ;

     x  :=  x  and  text_cursor_screen_mask  ;
     x  :=  x  xor  text_cursor_cursor_mask  ;

     char := x and $00FF ;
     attr := x shr 8 ;
end ;


procedure show_icon ;
var
   regs  :  Registers ;
   x,y   :  integer  ;
   c,a   :  byte  ;
begin
     if  display_text_icon  then  exit ;

     display_text_icon := true ;
     display_graph_icon := false;
     if  mouse_is_installed  then
     begin
          get_mouse_coordinate (x,y) ;
          x := x div 8 ;
          y := y div 8 ;

          set_cursor_to ( x,y ) ;
          GetChar ( old_char,old_attr ) ;  { save char at
icon }

          c := old_char ;
          a := old_attr ;
          mask_icon (c,a) ;

          PutChar ( c,a ) ;            { put new icon      }

          old_x := x ;                      { store icon
coordinate }
          old_y := y ;
     end {if} ;

end ;
```

```
procedure hide_icon ;
var
   regs  :  Registers ;
   x,y   :  integer  ;

begin
     if  not display_text_icon  then  exit ;

     display_text_icon := false ;
     if  mouse_is_installed  then
     begin
          set_cursor_to ( old_x,old_y ) ;         { delete
old icon }
          PutChar ( old_char,old_attr ) ;
     end {if} ;
end ;


procedure move_text_icon ;
var
   regs  :  Registers ;
   x,y   :  integer  ;
   c,a   :  byte     ;

          function  mouse_move : boolean ;
          begin
               mouse_move := not ( (x=old_x) and (y=old_y) )
;
          end ;
begin
     get_mouse_coordinate (x,y) ;
     x := x div 8 ;
     y := y div 8 ;

     if  mouse_move  then
     begin
          set_cursor_to ( old_x,old_y ) ;         { delete
old icon }
          PutChar ( old_char,old_attr ) ;

          set_cursor_to ( x,y ) ;
          GetChar ( old_char,old_attr ) ;  { save char at
icon }

          c := old_char ;
          a := old_attr ;
          mask_icon (c,a) ;

          PutChar ( c,a ) ;              { put new icon      }

          old_x := x ;                        { store icon
coordinate }
          old_y := y ;
     end {if} ;
```

```
end {move text icon} ;



function exist_file ( file_name : string ) : boolean ;
var
    information : SearchRec ;
    name        : string ;
const
     star = '*' ;
     ques = '?' ;
     NoError = 0 ;
begin
     name := file_name ;
     FindFirst ( name , Archive , information ) ;

     exist_file := (DosError = NoError)  and
                   (pos(star,name)=0)    and
                   (pos(ques,name)=0) ;
end ;



var  ViewPortSave : ViewPortType ;
     CP_save      : record  x,y : integer  end ;



Procedure  SaveGraphData ;
begin
     Cp_save.x := Graph.GetX  ;
     Cp_save.y := Graph.GetY  ;
     GetViewSettings ( ViewPortSave ) ;
     SetViewPort  ( 0,0 , GetMaxX,GetMaxY ,ClipOff ) ;
end ;

Procedure  RestoreGraphData ;
begin
     with  ViewPortSave  do  SetViewPort ( x1,y1 ,x2,y2
,Clip ) ;
     with  CP_save       do  MoveTo ( x,y ) ;
end ;
```

```
procedure  new_icon ( var icon : icon_type ) ;
begin
     if icon = nil then
       new (icon)
     else
       begin
          if icon^.shape1 <> nil then FreeMem (
icon^.shape1,icon^.size ) ;
          if icon^.shape2 <> nil then FreeMem (
icon^.shape2,icon^.size ) ;
       end
     {end if} ;

     icon^.size := 0 ;
     icon^.x1   := 0 ;  icon^.y1   := 0 ;
     icon^.x2   := 0 ;  icon^.y2   := 0 ;
     icon^.shape1 := nil ;
     icon^.shape2 := nil ;
end {new icon} ;



procedure  read_icon_from_file ( fname : string ; var icon :
icon_type ) ;
var
    data_file   : text ;
    i,j         : integer ;
    pixel       : word ;
    store       : ^byte ;

begin
     if not exist_file (fname) then exit ;

     SaveGraphData ;

     new_icon (icon) ;

     assign (data_file,fname) ;
     reset  (data_file) ;

     with  icon^  do
     begin
          readln ( data_file , x1,y1,x2,y2 ) ;

          size := ImageSize ( x1,y1,x2,y2 ) ;

          GetMem ( shape1 , size ) ;
          GetMem ( shape2 , size ) ;

          GetMem ( store , size ) ;
          GetImage ( 0,0,x2-x1,y2-y1 , store^ ) ;  { store
old picture on screen }
```

```
          for  j := 0 to y2-y1  do                { draw
screen mask to screen }
          for  i := 0 to x2-x1  do
          begin
               read ( data_file , pixel ) ;
               PutPixel ( i,j ,pixel ) ;
          end {double loop} ;
          GetImage ( 0,0,x2-x1,y2-y1 , shape1^ ) ;  { get
icon shpae }

          for  j := 0 to y2-y1  do                { draw
cursor mask to screen }
          for  i := 0 to x2-x1  do
          begin
               read ( data_file , pixel ) ;
               PutPixel ( i,j ,pixel ) ;
          end {double loop} ;
          GetImage ( 0,0,x2-x1,y2-y1 , shape2^ ) ;  { get
icon shpae }

          PutImage ( 0,0 , store^ , NormalPut ) ; { ReDraw
old picture }
          FreeMem  ( store , size ) ;

     end {with} ;

     close (data_file) ;

     RestoreGraphData ;

end ;


const  icon_in_used : icon_type = nil ;
       screen_store : ^byte     = nil ;


procedure  select_graph_icon ( var icon : icon_type ) ;
begin
     if  icon = nil  then  exit ;

     if  icon_in_used <> nil  then  { clear all setting from
last icon }
     begin
          FreeMem (screen_store , icon_in_used^.size) ;
     end {if} ;

     icon_in_used := icon ;

     GetMem (screen_store , icon_in_used^.size) ;
end ;
```

```
procedure show_graph_icon ;
var
   x,y  : integer ;
begin
     if  (display_graph_icon)
     or  (icon_in_used = nil)
     or  (icon_in_used^.shape1 = nil)   then  exit ;

     display_graph_icon := true ;
     display_text_icon  := false ;

     if  mouse_is_installed  then
     with  icon_in_used^  do
     begin
          SaveGraphData ;

          get_mouse_coordinate (x,y) ;

        if  (x+x1>=0) and (y+y1>=0)  then
          begin
               GetImage ( x+x1,y+y1, x+x2,y+y2 ,
screen_store^ ) ;  { store screen }

               PutImage ( x+x1,y+y1 ,shape1^ ,AndPut ) ;
{ draw icon    }
               PutImage ( x+x1,y+y1 ,shape2^ ,XorPut ) ;
          end {if} ;

          old_x := x ;                      { store icon
coordinate }
          old_y := y ;

          RestoreGraphData ;

     end {if with} ;
end ;


procedure hide_graph_icon ;
begin
     if  not display_graph_icon  then  exit ;

     display_graph_icon := false ;
     if  mouse_is_installed  then
     with  icon_in_used^  do
     begin
          SaveGraphData ;
          { delete old icon }
```

```
        if  (old_x+x1>=0) and (old_y+y1>=0)  then
           begin
                PutImage (old_x+x1,old_y+y1 ,screen_store^
,NormalPut);
           end {if} ;
           RestoreGraphData ;
     end {if} ;
end ;


procedure move_graph_icon ;
var
   x,y,dx,dy    :  integer  ;

           function  mouse_move : boolean ;
           begin
                mouse_move := not ( (x=old_x) and (y=old_y) )
;
           end ;

begin
     if  (not display_graph_icon)
     or  (icon_in_used = nil)
     or  (icon_in_used^.shape1 = nil)   then  exit ;

     get_mouse_coordinate (x,y) ;

     if  mouse_move  then
     with  icon_in_used^  do
     begin
           SaveGraphData ;

       if  (old_x+x1>=0) and (old_y+y1>=0)  then
           begin
                { delete old icon }
                PutImage (old_x+x1,old_y+y1 ,screen_store^
,NormalPut);
           end {if} ;

        if  (x+x1>=0) and (y+y1>=0)  then
           begin
                GetImage ( x+x1,y+y1, x+x2,y+y2 ,
screen_store^ ) ;{ store screen }

                PutImage ( x+x1,y+y1 ,shape1^ ,AndPut ) ;
{ draw icon     }
                PutImage ( x+x1,y+y1 ,shape2^ ,XorPut ) ;
           end {if} ;

           old_x := x ;                        { store icon
coordinate }
           old_y := y ;

           RestoreGraphData ;
```

```
      end {if with} ;

end {move graph icon} ;



procedure move_icon ;
begin
      if  mouse_is_installed  then
        if  display_text_icon   then
            move_text_icon
        else
            if  display_graph_icon  then
               move_graph_icon
            {end if}
        {end if} ;
      {end if}
end {move icon} ;


procedure set_mouse_pos ( x,y : integer ) ;
var
   regs  :  Registers ;
begin
      move_icon ;
      if  mouse_is_installed  then
      begin
           regs.AX := 4 ;
           regs.CX := 2 * x ;
           regs.DX := y ;
           intr ( mouse_int , regs ) ;
      end {if} ;
      move_icon ;
end ;


procedure get_mouse_pos ( var x,y : integer ) ;
var
   regs  :  Registers ;
begin
      if  mouse_is_installed  then
         begin
              move_icon ;
              get_mouse_coordinate (x,y) ;
         end
      else
         begin
              x := 0 ;
              y := 0 ;
         end
      {end if} ;
end ;
```

```
procedure set_button_wait_time ( time : integer ) ;
begin
     wait_time_for_next_presses := time {millisec} ;
end ;


procedure get_press_pos_of_button ( button_no : word ; var
x,y : integer );
begin
     move_icon ;
     x := x_pos_of_mouse_presses [button_no] ;
     y := y_pos_of_mouse_presses [button_no] ;
end ;


procedure clear_button_press ;
var
   regs  :  Registers ;
   count :  integer   ;
begin
     if  mouse_is_installed  then
        for count := 0 to 2 do
        with regs do
        begin
             move_icon ;
             AX := 5 ;
             BX := count ;
             intr ( mouse_int , regs ) ;
             x_pos_of_mouse_presses [count] := 0 ;
             y_pos_of_mouse_presses [count] := 0 ;
        end {for}
     {end if} ;
end ;


function  mouse_button_presses  ( button_no : byte ) :
integer ;
var
   regs  :  Registers ;
   count :  integer   ;
   again :  boolean   ;
begin
     count := 0 ;
```

```
     if  mouse_is_installed  then
     with  regs  do
     repeat
           move_icon ;
           AX := 5 ;
           BX := button_no ;
           intr ( mouse_int , regs ) ;
           again := ( BX > 0 ) ;        { BX return number of
times of persses }
           count := count + BX ;
           if again then
           begin
                x_pos_of_mouse_presses [button_no] := CX ;
                y_pos_of_mouse_presses [button_no] := DX ;
                delay ( wait_time_for_next_presses ) ;
           end {if} ;
     until  not again ;

     mouse_button_presses := count ;
end ;


procedure get_mouse_button_pressing ( var b : integer ) ;
var
   regs  :  Registers ;
begin
     if  mouse_is_installed  then
         with  regs  do
         begin
              move_icon ;
              AX := 3 ;
              intr ( mouse_int , regs ) ;
               b := BX ;
         end {with}
     else
         b :=  0 ;
     {end if} ;
end ;


function  pressing_mouse_button ( button_no : byte ) :
boolean ;
var  i,x,y : integer ;
     button_pressing : boolean ;
begin
     move_icon ;

     get_mouse_button_pressing ( x ) ;
     i := 1 shl button_no ;

     button_pressing :=  (x and i) = i ;
     pressing_mouse_button := button_pressing ;
```

```
     if  button_pressing  then
     begin
          get_mouse_pos ( x,y ) ;
          x_pos_of_mouse_presses [button_no] := x ;
          y_pos_of_mouse_presses [button_no] := y ;
     end ;
end ;


function  mouse_button_pressing : boolean ;
var  x : integer ;
begin
     get_mouse_button_pressing ( x ) ;
     mouse_button_pressing :=  (x<>0) ;
end ;


procedure set_mouse_bound ( x1,y1,x2,y2  : integer ) ;
var
   regs  :  Registers ;
begin
     if  mouse_is_installed  then
     with  regs  do
     begin
          move_icon ;
          if  (x1<0) or (x2<0) or (y1<0) or (y2<0)   then
exit ;

          AX := 7 ;
          CX := x1*2  ;   { multiply by 2 to report both odd
and even }
          DX := x2*2  ;
          intr ( mouse_int , regs ) ;
          AX := 8 ;
          CX := y1 ;
          DX := y2 ;
          intr ( mouse_int , regs ) ;
     end {with} ;
end ;


procedure set_Mickey_ratio ( x,y : word ) ;
var
   regs  :  Registers ;
begin
     if  mouse_is_installed  then
     with  regs  do
     begin
          move_icon ;
          AX := 15 ;
          CX := x  ;
          DX := y  ;
          intr ( mouse_int , regs ) ;
     end {if with} ;
```

```
end ;

procedure set_threshold ( x : word ) ;
var
   regs  :  Registers ;
begin
     if  mouse_is_installed  then
     with  regs  do
     begin
          move_icon ;
          AX := 19 ;
          DX := x  ;
          intr ( mouse_int , regs ) ;
     end {if with} ;
end ;



procedure read_mouse_motion ( var x,y : integer ) ;
var
   regs  :  Registers ;
begin
     if  mouse_is_installed  then
         with  regs  do
         begin
              move_icon ;
              AX := 11 ;
              intr ( mouse_int , regs ) ;
              x := CX div 2 ;
              y := DX ;
         end {with}
     else
         begin
              x := 0 ;
              y := 0 ;
         end
     {end if} ;
end ;


procedure define_text_cursor ( screen_mask , cursor_mask :
word ) ;
var
   regs  :  Registers ;
begin
     text_cursor_screen_mask  :=  screen_mask  ;
     text_cursor_cursor_mask  :=  cursor_mask  ;
```

```
     if  mouse_is_installed  then
     with  regs  do
     begin
          AX := 10 ;
          BX := 0 ;
          CX := screen_mask ;
          DX := cursor_mask ;
          intr ( mouse_int , regs ) ;
     end {with} ;
     move_icon ;
end ;

procedure wait_mouse_or_key  ;
begin
     repeat  move_icon  until  KeyPressed or
                    (mouse_button_presses (middle) > 0)  or
                    (mouse_button_presses (left)   > 0)  or
                    (mouse_button_presses (right)  > 0) ;
end ;



begin
     reset_mouse ;
     set_mouse_bound ( 0,0 ,639,199 ) ;
     define_text_cursor ( $B3FF , $3F00 ) ;
end ..
```

ภาคผนวก ก

ยูนิต music

ในยูนิตนี้มีคำสั่งที่น่าสนใจเพียงคำสั่งเดียวคือโพรซิเยอร์ music_call ซึ่งไม่มีพารามิเตอร์ ใช้สำหรับหยุดรอการกดคีย์ใดๆบนแป้นพิมพ์ หรือการปุ่มใดๆบนเมาส์ โดยจะส่งเสียงเตือนตลอดเวลา

โปรแกรมได้แสดงไว้ในหน้าถัดไป

```
unit music ;

interface

uses dos,crt,mouse ;

const  music_off = 0 ;
       music_on  = 1 ;

const  music_status : byte = music_on ;

procedure music_call ;


implementation

const  music_delay_time : integer = 75 ;

const {of note frequency}

C1 =  130  ;C1S = 138  ;D1 =  146  ;D1S = 154  ;E1 =  164
;F1 =  173  ;
F1S = 184  ;G1 =  195  ;G1S = 206  ;A1 =  219  ;A1S = 232
;B1 =  245  ;
C2 =  260  ;C2S = 276  ;D2 =  292  ;D2S = 309  ;E2 =  328
;F2 =  347  ;
F2S = 368  ;G2 =  389  ;G2S = 413  ;A2 =  437  ;A2S = 464
;B2 =  491  ;
C3 =  521  ;C3S = 552  ;D3 =  585  ;D3S = 619  ;E3 =  656
;F3 =  694  ;
F3S = 736  ;G3 =  780  ;G3S = 826  ;A3 =  876  ;A3S = 929
;B3 =  983  ;
C4 =  1042 ;

const { note of Sleepy lagoon }
     sleep : array [0..69] of integer =
     (a1s,2 ,c2,2, d2,2 ,f2,2 ,d3,3 ,
      c3,2 ,a2s,2 ,g2,2, f2,2 ,g2,3 ,
      f2,2 ,d2,2, c2,2 , a1s,2 ,d2,3 ,
      d2,4 ,d2,3, a1s,2 ,c2,2, d2,2
     ,f2,2 ,d3,3 ,c3,2 ,a2s,2 ,g2,2,
      f2,2 ,g2,3 ,f2,2 ,d2,2, c2,2 ,
      a1s,2 ,d2,3 ,d2,4 ,d2,3, 0,0 ) ;

const { note of H.M.Blue }
     HM_blue : array[0.. 135 ] of integer =
     (c2,1 ,c2,2 ,c2,1 ,a1s,2 ,a1,1 ,
      g1,2 ,g1,1 ,g1,5 ,g1,2 ,c2,1 ,
      c2,2 ,c2,1 ,a1s,2 ,a1,1 ,g1,2 ,
      d1s,1 ,e1,2 ,g1,1 ,a1,2 ,a1s,1,
      a1s,4 ,a1s,3 ,c2,2 ,d2,1 ,d2s,2 ,
      d2s,1 ,d2,2 ,c2,1 ,a1,2 ,c2,1 ,
      d2,2 ,d2s,1 ,d2s,4 ,d2s,2 ,c2,1 ,
```

```
       c2,2 ,c2,1 ,a1s,2 ,a1,1 ,g1,2 ,
       g1,1 ,g1,5 ,g1,2 ,g1,1 ,a1,2 ,
       g1,1 ,a1s,2 ,a1,1 ,g1,2 ,f1,1 ,
       f1,2 ,f1,1 ,g1,2 ,f1,1 ,g1s,2 ,
       g1,1 ,f1,2 ,e1,1 ,e1,2 ,e1,1 ,
       f1,2 ,f1s,1 ,g1,2 ,g1,1 ,d1s,2 ,
       d1,1 , c1,5 ,0,0 ) ;

const { note of Tis'sun down }
      sun_down : array[0..177 ] of real =
      (a1,3 ,c2,4 ,c2,4 ,c2,2.5 , c2,1 ,
       d2,2.5 , e2,1 ,d2,2.5 ,c2,1 ,a1,2.5 ,
       f1,1 ,g1s,4 ,g1s,4 ,g1,2.5 ,g1s,1,
       g1,2.5 ,f1,1 ,d1,3 ,c1,3 ,g1,4 ,
       g1,4 ,g1,2.5 ,g1s,1 ,g1,2.5 ,f1,1 ,
       d1,3 ,c1,3 ,f1,4 ,f1,4 ,

       a1,3 ,c2,4 ,c2,4 ,c2,2.5 , c2,1 ,
       d2,2.5 , e2,1 ,d2,2.5 ,c2,1 ,a1,2.5 ,
       f1,1 ,g1s,4 ,g1s,4 ,g1,2.5 ,g1s,1,
       g1,2.5 ,f1,1 ,d1,3 ,c1,3 ,g1,4 ,
       g1,4 ,g1,2.5 ,g1s,1 ,g1,2.5 ,f1,1 ,
       d1,3 ,c1,3 ,f1,4 ,f1,4 ,

       f1,4 ,f1,3 ,e1,3 ,a1,3 ,c2,3 ,
       e2,3 ,e2,4.5 ,e2,2 ,d2s,2 ,d2,3 ,
       c2,3 ,a1,3 , f1,3 ,g1s,4.5 ,g1s,2 ,
       f1,2 ,e1,3 ,g1s,3 ,b1,3 ,d2,3 ,
       d2,4.5 ,d2,2 ,e2,2 ,c2,3.5 ,a1,2 ,
       c2,3.5 ,d2,2 , a1s,3.5 ,g1,2 ,a1s,3.5 ,

       0,0);


procedure  play ( note : integer ; time  : real ) ;
var  i : integer ;
     dub : boolean ;
begin
     i :=  trunc( time ) ;
     dub := (time - i) <> 0.0 ;
     sound( note ) ;
     delay( music_delay_time * ( 1 shl i ) ) ;
     if  dub  then  delay( music_delay_time * ( 1 shl (i-1)
) ) ;
     nosound ;
end;
```

```
procedure  scan_key_to ( var scan_code , ascii_code  :  char
) ;
var  reg    : registers ;
begin
     wait_mouse_or_key ;
     if  keypressed  then
     begin
          { get key } reg.ah  :=  0 ; intr($16,reg) ;
          scan_code   :=  char(reg.ah) ;
          ascii_code  :=  char(reg.al) ;
     end ;
end ;

procedure music_call ;

var
   j : integer ;
   m : integer ;
   a,b : char ;
   not_loop : boolean ;
begin

     m := random (2) ;

     if  music_status = music_on  then
     case  m  of
           0  :  repeat
                      j := 0 ;
                      music_delay_time := 60 ;
                      repeat
                            play(trunc(sun_down[j]) ,
sun_down[j+1] ) ;
                            j := j + 2;
                            not_loop :=  keypressed
                                         or (
mouse_button_presses (middle)>0 ) ;
                     until  ((sun_down[j]=0) and
(sun_down[j+1]=0))
                             or not_loop ;
                until not_loop ;
```

```
          1   :  repeat
                          j := 0 ;
                          music_delay_time := 75 ;
                          repeat
                                play(HM_blue[j] , HM_blue[j+1]
) ;
                                j := j + 2;
                                not_loop :=  keypressed
                                          or (
mouse_button_presses (middle)>0 ) ;
                          until  ((HM_blue[j]=0) and
(HM_blue[j+1]=0))
                                  or not_loop ;
                   until not_loop ;
      end {case} ;

      scan_key_to (a,b) ;
end ;


begin { init section}
end..
```

ภาคผนวก ง

ยูนิต option

ในยูนิตนี้มีคำสั่งที่น่าสนใจเพียงคำสั่งเดียวคือโพรซิเยอร์ select_option ซึ่งไม่มีพารามิเตอร์ โพรซิเยอร์นี้จะทำหน้าที่แสดงเมนูออกมาให้เลือกดังในภาพที่ ก.9

โปรแกรมได้แสดงไว้ในหน้าถัดไป

```
unit  option ;

interface

uses crt,cell,glue4,menu,windsys,music,ploter ;

procedure select_option ;

implementation

const  option_wind : window_type = nil ;
       infor_wind  : window_type = nil ;

const  option_menu : menu_type   = nil ;

procedure music_switch ;
begin
     if music_status = music_off then
         music_status := music_on
     else
         music_status := music_off
     {end if} ;
end ;

procedure plotter_switch ;
begin
     if adeptor = plotter  then
        unselect_plotter
     else
        select_plotter
     {end if} ;
end ;

procedure select_option ;
var  again : boolean ;
     item  : byte ;
     inkey : char ;
begin

     open_window ( option_wind ) ;

     repeat
           new_menu( option_menu ) ;
```

```
            if  music_status = music_on  then
                add_menu ( option_menu ,'M',' Music ON    ')
            else
                add_menu ( option_menu ,'M',' Music OFF   ')
            {end if} ;

            if  adeptor = plotter  then
                add_menu ( option_menu ,'P',' Plotter ON  ')
            else
                add_menu ( option_menu ,'P',' Plotter OFF ')
            {end if} ;

            add_menu ( option_menu ,'O',' OS shell     ') ;
            add_menu ( option_menu ,'E',' Exit         ') ;

            clrscr ;
            select_menu ( option_menu ,item ,inkey ,{at} 1,1
) ;

            again := not ((inkey = esc) or ((inkey = return)
and (item = 4))
                          or (last_menu_scan_key=RightArrow)
                          or (last_menu_scan_key=LeftArrow)
);

            if  inkey = return   then
            begin
                 case  item  of
                         1  :  music_switch ;
                         2  :  plotter_switch ;
                         3  :  begin
                                    open_window( infor_wind )
;
                                    move_window_to ( -10,2 ,
relative ) ;
                                    clrscr;
                                    terminate_to_OS_shell ;
                                    clrscr;
                                    write('Press any key
...');
                                    wait ;
                                    close_window ;
                               end ;
                 end {case} ;
            end {if} ;


     until  not again ;

     close_window ;
end ;
```

```
begin
     music_status := music_on ;

     define_window  ( option_wind , 52,10 , 66,15 ) ;
     window_color   ( option_wind , LightGray , Blue ) ;
     broader_color  ( option_wind , White , Cyan ) ;
     window_heading ( option_wind , ' OPTIONS ') ;
     head_color     ( option_wind , Blue , Cyan ) ;

     define_window  ( infor_wind , 1,1 , 36,3 ) ;
     window_color   ( infor_wind , White , Red ) ;
     broader_color  ( infor_wind , Yellow , Red ) ;
     window_heading ( infor_wind , ' Remark ') ;
     head_color     ( infor_wind , LightGreen , Red ) ;

     new_menu  ( option_menu ) ;

     menu_normal_color    ( option_menu , Yellow ,Blue ) ;
     menu_inverse_color   ( option_menu , White , Magenta )
;
     menu_highlight_color ( option_menu , LightCyan ,Blue )
;
end .
```

ภาคผนวก น

ยูนิต ploter

ในการทำงานในลักษณะที่มีการใช้งานเครื่องพลอตเตอร์ (plotter) ในการนำภาพออกแสดงแทนจอภาพนั้น ได้สร้างคำสั่งขึ้นในยูนิตนี้ เพื่อจะเป็นคำสั่งที่ใช้ในการควบคุมและอำนวยความสะดวกทั้งหลายในการทำงานเกี่ยวกับพลอตเตอร์

<u>คำสั่งที่สร้างในยูนิตนี้</u>

ในยูนิตนี้มีคำสั่งที่เกี่ยวข้องกับการใช้งานพลอตเตอร์ดังนี้

1. โพรซิเยอร์ save_pen ไม่มีพารามิเตอร์ ใช้สั่งให้เครื่องพลอตเตอร์นำปากกามาเก็บไว้ยังที่เก็บปากกา

2. โพรซิเยอร์ set_plotter_port_to มีพารามิเตอร์หนึ่งตัว โพรซิเยอร์นี้จะใช้สำหรับกำหนดว่าจะให้ส่งคำสั่งควบคุมการทำงานของพลอตเตอร์ไปยังพอร์ตใด

3. โพรซิเยอร์ flush_plotter_buffer ไม่มีพารามิเตอร์ โพรซิเยอร์นี้จะใช้สำหรับสั่งให้ส่งคำสั่งควบคุมการทำงานของพลอตเตอร์ที่ยังค้างอยู่ออกไปให้หมด

4. โพรซิเยอร์ select_plotter ไม่มีพารามิเตอร์ โพรซิเยอร์นี้จะใช้เลือกอุปกรณ์ในการแสดงผลมาเป็นพลอตเตอร์

5. โพรซิเยอร์ unselect_plotter ไม่มีพารามิเตอร์ โพรซิเยอร์นี้จะใช้ยกเลิกการใช้พลอตเตอร์เป็นอุปกรณ์ในการแสดงผล

6. โพรซีเยอร์ graph_window, โพรซีเยอร์ plot, โพรซีเยอร์ draw, โพรซีเยอร์ GetMaxX, โพรซีเยอร์ GetMaxY, โพรซีเยอร์ GetMaxColor โพรซีเยอร์เหล่านี้จะใช้แทนที่คำสั่งทางกราฟฟิกเดิมซึ่งมีชื่อเดียวกัน แต่ไม่สามารถส่งภาพออกทางพลอตเตอร์ได้ คำสั่งใหม่นี้เมื่อถูกเรียกใช้จะไปดูว่า คำสั่ง select_ploter ถูกสั่งไว้หรือไม่ ถ้าถูกเลือกไว้ ก็จะส่งการวาดภาพออกไปทางเครื่องพลอตเตอร์ แต่ถ้าไม่ได้เลือกไว้ ก็จะส่งผลออกทางจอภาพตามปรกติ

โปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไป

```
unit  ploter ;

interface

uses crt,graph,glue4,cell ;

procedure save_pen ;

procedure graphwindow( x1,y1,x2,y2 : integer  ) ;

procedure plot ( x,y , color : integer )  ;

procedure draw ( x1,y1, x2,y2 , color : integer )  ;

procedure set_plotter_port_to ( a : string ) ;

procedure flush_plotter_buffer ;

procedure select_plotter ;

procedure unselect_plotter ;

function  GetMaxX : integer ;

function  GetMaxY : integer ;

function  GetMaxColor : integer ;

implementation


type
      paper_size    =  (A4,A3);
      coor          =  record  x,y : integer  end ;


const
      plotter_port : string =  'lpt2'  ;

      max_pen_plotter     =  8 ;

var
      HPGL     : text ;
      P1,P2    : coor ;
```

```
procedure set_plotter_port_to ( a : string ) ;
begin
     plotter_port := a ;
end ;


procedure  init_plotter ( paper : paper_size ) ;
begin
     assign  ( HPGL , 'NUL' ) ;
     rewrite ( HPGL ) ;

     case  paper  of
            A4   :  begin
                         P1.x  :=      0 ;
                         P1.y  :=      0 ;
                         P2.x  :=  10800 ;
                         P2.y  :=   7680 ;
                    end ;
            A3   :  begin
                         P1.x  :=      0 ;
                         P1.y  :=      0 ;
                         P2.x  :=  15200 ;
                         P2.y  :=  10800 ;
                    end ;
     end {case} ;

     graphwindow( 0,0 , GetMaxX,GetMaxY ) ;

end {init_plotter} ;

procedure  plot_edge( x1,y1,x2,y2 : real ) ;
const  comma = ',' ;
begin
     writeln(  HPGL , 'IP
',x1:6:0,comma,y1:6:0,comma,x2:6:0,comma,y2:6:0) ;
end;

procedure  plot_scale( x1,y1,x2,y2 : real ) ;
const  comma = ',' ;
begin
     writeln(  HPGL , 'SC
',x1:6:0,comma,x2:6:0,comma,y1:6:0,comma,y2:6:0) ;
end ;

procedure  plot_window( x1,y1,x2,y2 : real ) ;
const  comma = ',' ;
begin
     writeln(  HPGL , 'IW
',x1:6:0,comma,y1:6:0,comma,x2:6:0,comma,y2:6:0) ;
end ;
```

```
procedure  select_pen ( no : byte ) ;
begin
     if  no in [0..8]  then  writeln(  HPGL , 'SP', no )  ;
end ;

procedure  save_pen ;
begin
     writeln(  HPGL , 'SP 0' )  ;
end ;


procedure  pen_up ;
begin
     writeln( HPGL , 'PU' ) ;
end ;

procedure  pen_down ;
begin
     writeln( HPGL , 'PD' ) ;
end ;

procedure  move_to ( x,y : real) ;
const  comma = ',' ;
begin
     writeln( HPGL , 'PA ',x:6:0,comma,y:6:0 ) ;
end ;

procedure  move ( x,y : real) ;
const  comma = ',' ;
begin
     writeln( HPGL , 'PR ',x:6:0,comma,y:6:0 ) ;
end ;

procedure  line_type ( no : byte ; pitch : real ) ;
const  comma = ',' ;
begin
     if (no > -7) and (no < 7)  then
          writeln( HPGL , 'LT',no,comma,pitch )
     else
          writeln( HPGL , 'LT' )
     {end if} ;
end;

procedure  plotter_velocity ( x : integer ) ;
begin
     writeln( HPGL ,'VS',x );
end ;
```

```
{ new graphics command }



procedure  dot ( x,y , color : integer )  ;
begin
     select_pen (color) ;
     pen_up   ; move_to ( x,y ) ;
     pen_down ; pen_up  ;
end ;


procedure  line ( x1,y1, x2,y2 , color : integer )  ;
begin
     select_pen (color) ;
     pen_up   ; move_to ( x1,y1 ) ;
     pen_down ; move_to ( x2,y2 ) ;
     pen_up   ;
end ;


{plotter buffer system}


type
       dotptr     =  ^dot_data  ;
       dot_data   =  record
                          x,y  :  integer ;
                          ptr  :  dotptr  ;
                     end ;

       lineptr    =  ^line_data  ;
       line_data  =  record
                          x1,y1  :  integer ;
                          x2,y2  :  integer ;
                          ptr    :  lineptr  ;
                     end ;

const  full_dot   =  80 ;
       full_line  =  50 ;


var    line_stack :  array [0..max_pen_plotter] of
                     record
                          ptr    :  lineptr ;
                          count  :  byte    ;
                     end ;
```

```
       dot_stack  :  array [0..max_pen_plotter] of
                     record
                           ptr    :  dotptr ;
                           count  :  byte   ;
                     end ;

       adeptor_save : adeptor_type ;


procedure init_plot_buffer ;
var
    i   : integer  ;
begin
     for  i := 1 to max_pen_plotter do
     begin
          dot_stack[i].ptr    := nil ;
          dot_stack[i].count  := 0   ;
          line_stack[i].ptr   := nil ;
          line_stack[i].count := 0   ;
     end ;
end ;


procedure push_dot ( x,y , color : integer ) ;
var
   mem_need  : longint ;
   now,last  : dotptr ;
begin
     mem_need := 8 ;

     if dot_stack[color].ptr = nil then

       if maxavail > mem_need then
         begin
           new( dot_stack[color].ptr ) ;
           dot_stack[color].ptr^.x    := x ;
           dot_stack[color].ptr^.y    := y ;
           dot_stack[color].ptr^.ptr  := nil    ;
         end
       else
           out_of_memory
       {end if}

     else

       if maxavail > mem_need then
         begin
           now  := dot_stack[color].ptr ;
           while  now^.ptr <> nil  do  now := now^.ptr ;

           new( now^.ptr ) ;
           now := now^.ptr ;
           now^.x := x ;
```

```
            now^.y := y ;
            now^.ptr  := nil    ;
          end
        else
            out_of_memory ;
        {end if} ;
        dot_stack[color].count := dot_stack[color].count + 1
;


     {end if} ;

end {push_dot} ;


procedure clear_dot_buffer (color : integer ) ;
var
   now,next  : dotptr ;
begin
     next := dot_stack[color].ptr ;
     dot_stack[color].ptr := nil  ;

     if  next <> nil  then
     repeat
           now  := next ;
         . dot ( now^.x,now^.y , color ) ;
           next := now^.ptr ;
           dispose(now) ;
     until  next = nil ;
     dot_stack[color].count := 0 ;
end {new_menu};


procedure push_line ( x1,y1 ,x2,y2, color : integer ) ;
var
   mem_need  : longint ;
   now,last  : lineptr ;
begin
     mem_need := 12 ;

     if line_stack[color].ptr = nil then

       if maxavail > mem_need then
         begin
           new( line_stack[color].ptr ) ;
           line_stack[color].ptr^.x1   := x1 ;
           line_stack[color].ptr^.y1   := y1 ;
           line_stack[color].ptr^.x2   := x2 ;
           line_stack[color].ptr^.y2   := y2 ;
           line_stack[color].ptr^.ptr  := nil    ;
         end
       else
           out_of_memory
       {end if}
```

```
      else

        if maxavail > mem_need then
          begin
            now  := line_stack[color].ptr ;
            while  now^.ptr <> nil  do  now := now^.ptr ;

            new( now^.ptr ) ;
            now := now^.ptr ;
            now^.x1 := x1 ;
            now^.y1 := y1 ;
            now^.x2 := x2 ;
            now^.y2 := y2 ;
            now^.ptr  := nil    ;
          end
        else
            out_of_memory ;
        {end if} ;
        line_stack[color].count := line_stack[color].count +
1 ;

      {end if} ;

end {push_line} ;


procedure clear_line_buffer (color : integer ) ;
var
   now,next  : lineptr ;
begin
      next := line_stack[color].ptr ;
      line_stack[color].ptr := nil  ;

      if  next <> nil  then
      repeat
            now  := next ;
            line ( now^.x1,now^.y1 ,now^.x2,now^.y2 , color )
;
            next := now^.ptr ;
            dispose(now) ;
      until  next = nil ;
      line_stack[color].count := 0 ;
end {new_menu};


procedure  graphcolormode ;
begin
      if  adeptor <> plotter  then  glue1.graphcolormode
end;
```

```
procedure plot( x,y , color : integer ) ;
begin
     if  adeptor = plotter   then
     begin
       if color > 0 then
       begin
            color := ((color-1) mod max_pen_plotter) + 1 ;
            push_dot ( x,y, color ) ;
            if dot_stack[color].count = full_dot  then
                clear_dot_buffer (color)  ;
       end {if}
     end
     else
        glue4.plot ( x,y , color )
     {end if} ;
end {plot} ;


procedure draw( x1,y1 ,x2,y2, color : integer ) ;
begin
     if  adeptor = plotter  then
     begin
       if color > 0 then
       begin
            color := ((color-1) mod max_pen_plotter) + 1 ;
            push_line ( x1,y1 , x2,y2, color ) ;
            if line_stack[color].count = full_line  then
                clear_line_buffer (color)
            {end if} ;
       end
     end
     else
         glue4.draw ( x1,y1 ,x2,y2 , color )
     {end if} ;

end {draw} ;


function  GetMaxX ;
begin
     if  adeptor = plotter   then
       GetMaxX := 640
     else
       GetMaxX := Graph.GetMaxX
     {end if} ;
end ;
```

```
function  GetMaxY ;
begin
     if  adeptor = plotter   then
       GetMaxY := 350
     else
       GetMaxY := Graph.GetMaxY
     {end if} ;
end ;


function  GetMaxColor : integer ;
begin
     if  adeptor = plotter   then
       GetMaxColor := max_pen_plotter
     else
       GetMaxColor := Graph.GetMaxColor
     {end if} ;
end ;




(*  select between plotter & monitor *)


procedure SetTextStyle( Font,Direction : word ; CharSize :
word ) ;
begin
     case  adeptor  of
        plotter : begin { do nothing } end ;
     else
          Graph.SetTextStyle( Font,Direction ,Charsize ) ;
     end {case}
end {draw} ;

procedure SetTextJustify ( Horiz,Vert : word ) ;
begin
     case  adeptor  of
        plotter : begin { do nothing } end ;
     else
          graph.SetTextJustify ( Horiz,Vert ) ;
     end {case}
end {draw} ;
```

```
procedure SetUserCharSize ( MulX,DivX,MulY,DivY : word ) ;
begin
     case  adeptor  of
        plotter : begin { do nothing } end ;
     else
         graph.SetUserCharSize ( MulX,DivX,MulY,DivY ) ;
     end {case}
end {draw} ;



procedure SetColor ( Color : word ) ;
begin
     case  adeptor  of
        plotter : begin { do nothing } end ;
     else
         graph.SetColor( Color ) ;
     end {case}
end {draw} ;


procedure graphwindow( x1,y1,x2,y2 : integer  ) ;

          procedure swap ( var a,b : integer ) ;
          var temp : integer ;
          begin
               temp := a ; a := b ; b := temp ;
          end ;

var
   newP1  ,
   newP2  : coor ;
begin
     case  adeptor  of
        plotter : begin
                  flush_plotter_buffer ;

                  newP1.x  :=  P1.x + round ( (P2.x-P1.x) *
(x1 / GetMaxX)) ;
                  newP1.y  :=  P2.y - round ( (P2.y-P1.y) *
(y1 / GetMaxY)) ;
                  newP2.x  :=  P1.x + round ( (P2.x-P1.x) *
(x2 / GetMaxX)) ;
                  newP2.y  :=  P2.y - round ( (P2.y-P1.y) *
(y2 / GetMaxY)) ;

                  plot_edge   ( newP1.x,newP1.y
,newP2.x,newP2.y ) ;

                  plot_window ( newP1.x,newP1.y
,newP2.x,newP2.y)  ;
```

```
                plot_scale  ( 0,0, x2-x1,y2-y1 ) ;
              end ;
     else
         glue4.graphwindow ( x1,y1 ,x2,y2 ) ;
     end {case} ;

end ;



procedure  flush_plotter_buffer ;
var i : byte ;
begin
     for i := 1 to max_pen_plotter  do
     begin
          clear_dot_buffer (i) ;
          clear_line_buffer (i) ;
     end ;
     save_pen ;
     flush ( HPGL ) ;
end ;


procedure  select_plotter ;
begin
     close (HPGL) ;
     assign( HPGL , plotter_port ) ;
     rewrite ( HPGL ) ;

     if adeptor <> plotter then adeptor_save := adeptor ;
     adeptor       := plotter ;

     graphwindow( 0,0 , GetMaxX,GetMaxY ) ;
end ;

procedure  unselect_plotter ;
begin
     close ( HPGL );
     assign( HPGL , 'NUL' ) ;
     rewrite ( HPGL ) ;
     if adeptor = plotter then  adeptor := adeptor_save ;
end ;



begin  { init section }

     initgraphic ;
     init_plotter ( A4 ) ;
     init_plot_buffer ;
     adeptor_save := adeptor ;

     pen_up ;
end..
```

ภาคผนวก ข

ยูนิต timing

คำสั่งที่สร้างขึ้นในยูนิตนี้จะเป็นคำสั่งที่ใช้ในเรื่องเกี่ยวกับเวลาและการจับเวลา

<u>คำสั่งที่สร้างในยูนิต</u>

ในยูนิตนี้มีคำสั่งที่น่าสนใจดังนี้

1. ฟังชัน second ไม่มีพารามิเตอร์ จะส่งเวลาเป็นจำนวนวินาที โดยนับตั้งแต่เวลา 00.00 น.ผ่านกลับทางชื่อของฟังชัน

2. ฟังชัน base_60 จะหน้าที่แปลงเวลาจากวินาทีที่เป็นจำนวนจริงไปเป็น ชั่วโมง นาที วินาที แล้วส่งผลกลับผ่านกลับทางชื่อของฟังชัน

3. ฟังชัน time จะส่งเวลาเป็น ชั่วโมง นาที วินาที กลับผ่านกลับทางชื่อของฟังชัน

4. โพรซีเยอร์ start, โพรซีเยอร์ stop, ตัวแปร time_used จะใช้ในการจับเวลา โดยที่ ในโปรแกรมที่ต้องการจับเวลาจะต้องใช้คำสั่ง start ก่อน และเมื่อตามด้วยคำสั่ง stop แล้ว ค่าเวลาที่ใช้ไปเป็นวินาทีระหว่างคำสั่ง start กับ stop จะบันทึกอยู่ในตัวแปร time_used

ตัวโปรแกรมได้แสดงไว้โดยละเอียดในหน้าก่อนไปแล้ว

```
unit timing ;

(*                                  *)
(*      Filename  :  TIMING         *)
(*                                  *)

interface

uses   dos ;

var    time_used  :  real  ;

function  second : real ;
function  base_60 ( x : real ) : string ;

function  time  : string ;
procedure start ;
procedure stop  ;



implementation



function second : real ;
var
     hour , minute , sec , sec100  :  word ;
begin
     GetTime( hour , minute , sec , sec100 ) ;
     second := hour * 3600.0  + minute * 60.0 + sec +
(sec100 / 100.0) ;
end {function second} ;


function  base_60 ( x : real ) : string ;
const
       colon = ':' ;
       dot   = '.' ;
var
     a,b,t  : string[11] ;
     y,z,
     dummy  : integer ;
```

```
          function two_digit( x : integer ) : string ;
          var   temp : string[8] ;
                l    : integer ;
          begin
               str ( x , temp ) ;
               temp := '000' + temp  ;
               l := length (temp) ;
               two_digit := copy(temp,l-1,2);
          end ;

begin
     a := '' ;
     y := trunc( frac(x) * 100 ) ;
     a := dot + two_digit ( y ) ;
     x := x / 60.0 ;
     y := trunc( frac(x) * 60 ) ;
     a := colon + two_digit ( y ) + a ;
     y := trunc(x) ;
     a := colon + two_digit ( y mod 60  ) + a  ; y :=  y div
60  ;
     a := two_digit (y) + a ;
     base_60 := a ;
end {base_60} ;



function  time  :  string ; begin time := base_60 (second) ;
end ;
procedure start ;  begin  time_used := second  end ;
procedure stop  ;  begin  time_used := second - time_used
end ;


begin {init section}

end ..
```

ภาคผนวก ง

ยูนิต vardef

ยูนิตนี้ใช้ในการเตรียมข้อมูลบางประเภทที่มีการเรียกใช้งานอย่างทั่วไปภายในโปรแกรม

สิ่งที่กำหนดขึ้นในยูนิตนี้

ในยูนิตนี้มีค่าคงที่ที่สำคัญคือ max_data ซึ่งใช้กำหนดขนาดของข้อมูลที่จะนำมาคำนวณ

การในยูนิตนี้มีการเตรียมประเภทของข้อมูลที่สำคัญอยู่สองประเภทด้วยกันคือ

1. ประเภท vector ซึ่งจะใช้กำหนดตัวแปรชนิดแถวลำดับของจำนวนเชิงซ้อน

2. ประเภท matrix จะใช้กำหนดตัวแปรชนิดเมทริกซ์ที่มีสมาชิกเป็นจำนวนเชิงซ้อน

คำสั่งที่น่าสนใจในยูนิตนี้มีดังนี้

1. โพรซีเยอร์ new_matrix มีหน้าที่เตรียมเนื้อที่ให้เป็นเมทริกซ์ ใช้สำหรับเตรียมตัวแปรที่มีประเภทเป็น matrix เนื่องจากตัวแปรประเภทนี้จะต้องมีการจองเนื้อที่ในหน่วยความจำเป็นการล่วงหน้า ซึ่งทำได้โดยใช้คำสั่งนี้

2. โพรซีเยอร์ cancel_matrix มีหน้าที่เตรียมเนื้อที่ให้เป็นเมทริกซ์ ใช้สำหรับยกเลิกการจองเนื้อที่ของตัวแปรที่มีประเภทเป็น matrix เนื่องจากว่าตัวแปรประเภทนี้จะต้องมีการ

ของเนื้อที่ในหน่วยความจำเป็นการต่างหาก และเมื่อหมดความจำเป็น ก็สามารถเรียกหน่วยความจำคืนมาเพื่อใช้ในงานอื่นๆได้ โดยใช้คำสั่ง

ตัวแปรแกรมได้แสดงไว้โดยละเอียดในหัวข้อที่ผ่านมาแล้ว

```
unit vardef ;

interface

uses  complx  {unit of complex variable and function }  ;

const
     max_data        =   32  ;
     bell            =   #07 ;
     esc             =   #27 ;
     ret             =   #13 ;
     space           =   #32 ;
     null            =   ''  ;

     minimum_real    =  -1.0E+37   ;
     maximum_real    =  +1.0E+37   ;

type
     array_of_complex  =  array [0..max_data] of complex ;

     vector            =  array_of_complex ;

     array_of_vector   =  array [0..max_data] of ^vector ;

     matrix            = ^array_of_vector ;


(*
      To declare variable of type 'vector' and 'matrix' use
form

      VAR     <vardef>  :  VECTOR  ;

      and

      CONST   <vardef>  :  MATRIX  =  NIL  ;

      then  use procedure 'new_matrix' to  reserve  memory
for matrix .

      To destroy them  use procedure 'cancel_matrix' .

*)


var
     signal_data_file_name : string ;
```

```
procedure  new_matrix    ( var u : matrix ) ;

procedure  cancel_matrix ( var u : matrix ) ;


implementation


{$ifdef  cpu87}
  const  real_length = 8 ;
{$else}
  const  real_length = 6 ;
{$endif}



procedure  new_matrix ( var u : matrix ) ;
var  i          : integer ;
     mem_need : longint ;
begin
     if  u = nil  then
     begin
          new (u) ;
          mem_need := 2 * max_data * real_length ;

          for  i := 0 to max_data  do
          begin

               if maxavail > mem_need then
                  new ( u^[i] )
               else
                 begin
                   writeln (' ****  THERE ARE NOT ENOUGTH
MENORY FOR MATRIX  ****');
                   halt ;
                 end
               {end if} ;

          end {loop} ;
     end {if} ;
end {new_matrix} ;
```

```
procedure  cancel_matrix ( var u : matrix ) ;
var  i : integer ;
begin
     if  u <> nil  then
     begin
          for i := 0 to max_data do
          begin
               dispose (u^[i]) ;
          end {loop} ;

          dispose (u) ;
          u := nil ;
     end {if} ;
end ;



begin  { init section  }

end.
```

ภาคผนวก ฆ

ยูนิต windsys

ในการทำงานในลักษณะที่มีการใช้งานหน้าต่าง (window) เพื่อช่วยให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้สะดวกขึ้น และทำให้ผู้ใช้เข้าใจวิธีใช้งานได้ง่ายขึ้นนั้น ได้สร้างคำสั่งขึ้นในยูนิตนี้ เพื่อจะเป็นคำสั่งที่ใช้ในการควบคุมและอำนวยความสะดวกในการทำงานเกี่ยวกับหน้าต่างทั้งหลาย

ข้อมูลและคำสั่งที่สร้างในยูนิต

ในยูนิตนี้ได้มีการกำหนดประเภทของตัวแปรขึ้นมาใหม่ประเภทหนึ่ง ซึ่งหากจะใช้งานคำสั่งเกี่ยวกับหน้าต่างที่สร้างขึ้นแล้ว ก็จำเป็นต้องทราบเกี่ยวกับตัวแปรประเภทนี้พอสมควร ประเภทของตัวแปรที่กำหนดขึ้นมานี้คือ window_type ซึ่งจะใช้ในการเก็บข้อมูลเกี่ยวกับหน้าต่างทั้งหลาย และโดยที่หน้าต่างอาจมีขนาดหน่วยความจำได้ไม่แน่นอน ดังนั้น จึงได้กำหนดให้ตัวแปรประเภทนี้เป็นตัวแปรพลวัต (dynamic variable) ซึ่งสามารถโปรแกรมให้ขนาดของตัวแปรมีขนาดแปรไปได้ตามต้องการ

ในยูนิตนี้มีคำสั่งที่เกี่ยวข้องกับการใช้งานหน้าต่างดังนี้

1. โพรซิเยอร์ define_window มีพารามิเตอร์ห้าตัวด้วยกัน ตัวแรกเป็นประเภท window_type ซึ่งโพรซิเยอร์นี้จะทำหน้าที่จองเนื้อที่ในหน่วยความจำ และเตรียมข้อมูลเกี่ยวกับหน้าต่างลงในตัวแปรนี้ให้พร้อมที่จะนำไปใช้งาน แล้วจึงส่งกลับ ส่วนพารามิเตอร์สี่ตัวหลังใช้กำหนดขนาดของหน้าต่างที่จะสร้างขึ้น

2. โพรซิเยอร์ cancel_window มีพารามิเตอร์หนึ่งตัว เป็นประเภท

window_type ซึ่งโพรซิเยอร์นี้จะทำหน้าที่ยกเลิกข้อมูลและหน่วยความจำที่จองไว้ภายในตัวแปรนี้แล้วจึงส่งกลับ

3. โพรซิเยอร์ clean_window มีพารามิเตอร์หนึ่งตัว โพรซิเยอร์นี้จะใช้ลบข้อมูลที่อยู่ในหน่วยความจำที่จองไว้สำหรับหน้าต่าง

4. โพรซิเยอร์ window_heading จะใช้ในการกำหนดข้อความที่จะปรากฏที่ขอบบนของช่องหน้าต่าง

5. โพรซิเยอร์ bottom_line จะใช้ในการกำหนดข้อความที่จะปรากฏที่ขอบล่างของช่องหน้าต่าง

6. โพรซิเยอร์ broader_and_head จะใช้ในการนำข้อความที่กำหนดไปปรากฏที่ขอบบนและขอบล่างของช่องหน้าต่าง

7. โพรซิเยอร์ broader_color, โพรซิเยอร์ head_color, โพรซิเยอร์ bottom_color ทั้งสามโพรซิเยอร์นี้จะใช้ในการกำหนดสีที่จะใช้ในขณะนำเมนูออกแสดง โดยจะใช้กำหนดสีของ กรอบหน้าต่าง, ข้อความที่ขอบบน, ข้อความที่ขอบล่าง ตามลำดับ

8. โพรซิเยอร์ open_window ใช้สำหรับเปิดหน้าต่างตามที่ได้กำหนดไว้โดยคำสั่ง define_window ขึ้นบนจอภาพ

9. โพรซิเยอร์ close_window ใช้สำหรับปิดหน้าต่างตามที่ได้กำหนดไว้โดยคำสั่ง open_window ครั้งท้ายสุดออกจากจอภาพ

10. โพรซิเยอร์ move_window ใช้สำหรับเลื่อนหน้าต่างตามที่ได้กำหนดไว้โดยคำสั่ง open_window ครั้งท้ายสุด ไปในทิศทางที่กำหนดโดยพารามิเตอร์ตัวที่หนึ่ง เป็นระยะทางเท่ากับที่กำหนดโดยพารามิเตอร์ตัวที่สอง

11. โพรซิเยอร์ move_window_to ใช้สำหรับเลื่อนหน้าต่างตามที่ได้กำหนดไว้โดยคำสั่ง open_window ครั้งท้ายสุด ไปยังตำแหน่งที่กำหนดโดยพารามิเตอร์ตัวสองตัวแรก

ในการนำคำสั่งเหล่านี้ไปใช้ช่วยในการเขียนโปรแกรม จะเริ่มที่ผู้เขียนจะต้องกำหนดตัวแปรขึ้นมาโดยให้มีประเภทเป็น window_type แล้วจึงสั่ง define_window จากนั้นจึงเพิ่มรายละเอียดเข้าไปทีละรายการโดยคำสั่งต่างๆ เมื่อเรียบร้อยแล้วจึงนำตัวแปรที่ได้ไปใช้โดยคำสั่ง open_window หรือคำสั่งอื่นๆต่อไป

โปรแกรมได้แสดงไว้โดยละเอียดในหน้าถัดไป

```
unit windsys ;

interface

uses   dos,crt,cell ;

const
      of_screen        =  0 ;
      relative         =  1 ;
      of_last_window   =  1 ;

type
      intptr           =  ^integer ;
      directional      =  (up,right,down,left) ;
      window_type      =  ^window_status ;
      window_status    =  record
                              window_ptr  :  intptr ; {
pointer to buffer }
                              window_edge :  record

x1,y1,x2,y2 : byte ;
                                             end ;
                              cursor_pos  :  record  x,y :
byte ; end ;
                              buffer_size :  integer ;
                              header      :  string ;
                              bottom      :  string ;
                              on_screen   :  boolean  ;
                              head_attr    : byte ;
                              bott_attr    : byte ;
                              broader_attr : byte ;
                              window_attr  : byte ;
                              pred_window :  window_type ;
{ point to last }

{ window         }
                         end ;


procedure  define_window  ( var  wind : window_type ; x1,y1
, x2,y2 : byte ) ;
procedure  cancel_window  ( var  wind : window_type ) ;
procedure  clean_window   ( wind : window_type ) ;
procedure  window_heading ( wind : window_type ; a :  string
) ;
procedure  bottom_line    ( wind : window_type ; a :  string
) ;
procedure  broader_and_head ( wind : window_type ) ;
procedure  broader_color  ( wind : window_type ; fore , back
: byte ) ;
procedure  head_color     ( wind : window_type ; fore , back
: byte ) ;
procedure  bottom_color   ( wind : window_type ; fore , back
: byte ) ;
```

```
procedure  open_window    ( wind : window_type ) ;
procedure  move_window    ( direction : directional ; length
: integer ) ;
procedure  move_window_to ( x,y : integer ; mode : byte ) ;

procedure  window_color     ( wind : window_type ; fore ,
back : byte ) ;

procedure  close_window   ;


procedure  scan_key_to ( var scan_code , ascii_code  :  char
) ;

function   ReadKey : char ;


implementation

const
      max_line     =  25 ;
      max_col      =  80 ;
      null         =  '' ;

type
      screen_type = (CGA,EGA,Hercules) ;
      screen_area = array [ 1..max_line, 1..max_col ] of
integer ;
      screen_ptr  = ^screen_area ;

var
      main_screen     :  window_type   ;
      window_in_used  :  window_type   ;

procedure  swapc ( var a , b ) ;
var temp : byte ;
    x    : byte absolute a ;
    y    : byte absolute b ;
begin
     temp := x ; x := y ; y := temp ;
end ;


procedure set_screen_ptr ( var scrptr : screen_ptr ) ;
var  mode , page , col : byte ;
begin
     get_display_mode ( mode , page , col ) ;
     case  mode  of
         $02,$03 :  scrptr := ptr ( $B800 ,
(max_col*max_line) * page ) ;
           $07   :  scrptr := ptr ( $B000 ,
(max_col*max_line) * page ) ;
```

```
        else
             scrptr := ptr ( $B000 , $0000 ) ;
     end {case} ;
end {set_screen_ptr} ;


procedure broader ( x1,y1, x2,y2 , attr : integer ) ;
const
     min = #176 ;  mid = #177 ;  max = #178 ;
     ulc = #$C9 ;  us  = #$CD ;  urc = #$BB ;
     ls  = #$BA ;                rs  = #$BA ;
     llc = #$C8 ;  ds  = #$CD ;  lrc = #$BC ;
var
     i  :  integer ;
     attr_save : byte ;
begin
     swapc ( attr , TextAttr ) ;
     gotoxy(x1,y1) ; OutChar (mid{ulc}) ;
     for i := x1+1 to x2-1 do  begin gotoxy(i,y1) ;
OutChar(mid) ; end ;
     gotoxy(x2,y1) ; OutChar (mid{urc}) ;
     for i := y1+1 to y2-1 do
     begin
          gotoxy(x1,i) ; OutChar(ls) ;
          gotoxy(x2,i) ; OutChar(rs) ;
     end;
     gotoxy(x1,y2) ; OutChar (llc) ;
     for i := x1+1 to x2-1 do  begin gotoxy(i,y2) ;
OutChar(ds) ; end ;
     gotoxy(x2,y2) ; OutChar (lrc) ;
     swapc ( attr , TextAttr ) ;
end;

procedure init_window_system ;
var  mem_need : word ;
begin
     mem_need := SizeOf (main_screen^)  ;

     if MaxAvail > mem_need then
      begin
          new (main_screen) ;
          window_in_used  :=  main_screen ;

          with main_screen^ do
          begin
               window_edge.x1  :=  1  ;    window_edge.y1
:=  1  ;
               window_edge.x2  :=  80 ;    window_edge.y2
:=  25 ;
               cursor_pos.x := wherex ;    cursor_pos.y :=
wherey ;
               window_attr  := TextAttr ;
               pred_window  := nil ;
          end {with} ;
```

```
      end
     else
      out_of_memory
    {end if}
end {init_window_system} ;


procedure clean_window ( wind : window_type ) ;
var
    i       ,
    segment,
    offset : word ;
    pointer : intptr ;
begin
     if  (wind <> nil) and (not wind^.on_screen)  then
     with  wind^  do
     begin
          cursor_pos.x  :=  1 ;
          cursor_pos.y  :=  1 ;

          segment := seg(window_ptr^) ;
          offset  := ofs(window_ptr^) ;

          for i :=  1 to (buffer_size div 2) do
          begin
               pointer   :=   ptr ( segment , offset ) ;
               pointer^  :=   integer((TextAttr * $100) +
$20)   ;  { $20 = space }
               offset    :=   offset + 2 ;
          end {loop} ;

     end {if with} ;
end {clean_window} ;


procedure  define_window  ( var wind : window_type ; x1,y1 ,
x2,y2 : byte ) ;
var  buff_need , mem_need : word ;
begin

     if (wind = nil) then
     begin
          if y1 < 1          then  y1 := 1 ;
          if y2 > max_line   then  y2 := max_line ;
          if x2 > max_col    then  x2 := max_col ;
          if x1 < 1          then  x1 := 1 ;

          buff_need :=  2 * (x2-x1+1) * (y2-y1+1) ;
          mem_need  :=  SizeOf(wind^) + buff_need ;

          if  MaxAvail < mem_need  then  out_of_memory ;

          new ( wind ) ;
```

```
          with  wind^  do
          begin
               header       := null ;
               bottom       := null ;
               on_screen    := false ;

               window_edge.x1   :=  x1 ; window_edge.y1   :=
y1 ;
               window_edge.x2   :=  x2 ; window_edge.y2   :=
y2 ;

               cursor_pos.x   :=  1 ;    cursor_pos.y   :=  1
;

               window_attr    :=  TextAttr ;
               broader_attr   :=  TextAttr ;
               head_attr      :=  TextAttr ;
               bott_attr      :=  TextAttr ;

               buffer_size    :=  buff_need ;
               getmem( window_ptr, buffer_size ) ;
          end {with} ;
          clean_window ( wind ) ;
     end {if} ;
end {define_window} ;


procedure  window_heading ( wind : window_type ; a :  string
) ;
begin
     wind^.header := a ;
end ;

procedure  bottom_line ( wind : window_type ; a :  string )
;
begin
     wind^.bottom := a ;
end ;

procedure  broader_color ( wind : window_type ; fore , back
: byte ) ;
begin
     back := back mod 8 ;
     wind^.broader_attr := back*16 + fore ;
     wind^.head_attr    := back*16 + fore ;
     wind^.bott_attr    := back*16 + fore ;
end ;

procedure  window_color ( wind : window_type ; fore , back :
byte ) ;
begin
     back := back mod 8 ;
     wind^.window_attr := back*16 + fore ;
end ;
```

```
procedure  head_color ( wind : window_type ; fore , back :
byte ) ;
begin
     back := back mod 8 ;
     wind^.head_attr := back*16 + fore ;
end ;
procedure  bottom_color ( wind : window_type ; fore , back :
byte ) ;
begin
     back := back mod 8 ;
     wind^.bott_attr := back*16 + fore ;
end ;


procedure  cancel_window ( var wind : window_type ) ;
var
     i : integer ;
begin
     if (wind <> nil) and (not wind^.on_screen) then
     begin
          freemem( wind^.window_ptr , wind^.buffer_size ) ;
          dispose( wind ) ;
          wind := nil ;
     end ;
end {cancel_window};



procedure  swap_screen_with_buffer( wind : window_type ) ;
var
     screen  :  screen_ptr ;

     pointer :  intptr ;
     line    ,
     linesize,
     segment ,
     offset  :  word ;
     temp    :  array [1..max_col] of integer ;
begin
     set_screen_ptr ( screen ) ;

     if  wind <> nil  then
     with  wind^  do
     begin
          segment := seg(window_ptr^) ;
          offset  := ofs(window_ptr^) ;

          with window_edge do
          begin
               linesize :=  (x2-x1+1)*2 ;
```

```
                for line :=  y1 to y2 do
                begin
                     pointer   :=  ptr ( segment , offset ) ;
                     move ( pointer^ , temp , linesize ) ;
                     move ( screen^[line,x1] , pointer^ ,
linesize )  ;
                     move ( temp , screen^[line,x1] ,
linesize ) ;
                     offset  :=  offset + linesize ;
                end {loop} ;
           end {with window_edge} ;
     end {if with} ;
end {swap_screen_with_buffer} ;

procedure  broader_and_head ( wind : window_type ) ;
var  i , head_length  :  shortint ;
     xo,yo : byte ;
begin
     with  wind^  do
     with  window_edge do
     begin
          xo  :=  whereX ;  yo  :=  whereY ;
          window (1,1,max_col,max_line) ;
          broader (x1,y1,x2,y2 , broader_attr ) ;

          swapc ( head_attr , TextAttr ) ;
          head_length := length ( header ) ;

          if head_length > 0 then
          begin
               i := ( (x2-x1+1) - head_length ) div 2 ;
               if i < 1 then  i := 1 ;
               gotoxy( x1+i, y1 ) ;
               OutString (copy(header,1,(x2-x1-1)));
          end ;
          swapc ( head_attr , TextAttr ) ;

          swapc ( bott_attr , TextAttr ) ;
          if length( bottom ) > 0 then
          begin
               gotoxy( x1+1 ,y2 ) ;
               OutString (copy(bottom,1,(x2-x1-1)));
          end ;
          swapc ( bott_attr , TextAttr ) ;
          window ( x1+1,y1+1 , x2-1,y2-1 ) ;
          gotoxy( xo , yo ) ;
     end {with} ;
end {procedure head_and_bottom} ;
```

```
procedure  open_window ( wind : window_type ) ;
begin
     if  (wind <> nil) and (not wind^.on_screen)  then
     with  wind^  do
     begin
          pred_window    :=  window_in_used ;
          window_in_used :=  wind {opening now} ;

          with  pred_window^  do
          begin
               cursor_pos.x  :=  whereX ;
               cursor_pos.y  :=  whereY ;
               window_attr   :=  TextAttr ;
          end ;

          TextAttr := window_attr ;

          swap_screen_with_buffer( wind ) ;

          wind^.on_screen  :=  true ;

          broader_and_head (wind) ;
          gotoxy( cursor_pos.x , cursor_pos.y ) ;

     end {with wind^ } ;

end ;

procedure  close_window  ;
begin
     if  window_in_used <> main_screen then
     begin
          with  window_in_used^  do
          begin
               cursor_pos.x  :=  whereX ;
               cursor_pos.y  :=  whereY ;
               window_attr   :=  TextAttr ;
               on_screen     :=  false ;
               swap_screen_with_buffer( window_in_used ) ;
               window_in_used := pred_window  ;
          end {with} ;

          with  {new} window_in_used^  do
          begin
               with  window_edge  do
               if  window_in_used = main_screen  then
                 window ( x1,y1 , x2,y2 )
               else
                 window ( x1+1,y1+1 , x2-1,y2-1 )
               {end if} ;
               TextAttr := window_attr ;
               gotoxy( cursor_pos.x , cursor_pos.y ) ;
          end {with} ;
     end {if} ;
```

```
end ;


procedure  move_window( direction : directional ; length :
integer ) ;

            procedure  swap ;
            begin
                 swap_screen_with_buffer( window_in_used ) ;
            end {swap} ;

var no : integer ;
begin
     if  window_in_used <> main_screen  then
     begin
          with  window_in_used^  do
          begin
               cursor_pos.x := WhereX ;
               cursor_pos.y := WhereY ;
          end {with} ;

          with  window_in_used^.window_edge  do
          case  direction  of
                    up   :  if   y1-length >= 1  then
                            begin
                                 swap ;
                                 y1 := y1-length ;
                                 y2 := y2-length ;
                                 swap;
                            end ;
                  down   :  if   y2+length <= max_line-1
then
                            begin
                                 swap;
                                 y1 := y1+length ;
                                 y2 := y2+length ;
                                 swap;
                            end ;
                 right   :  if   x2+length <= max_col  then
                            begin
                                 swap;
                                 x1 := x1+length ;
                                 x2 := x2+length ;
                                 swap;
                            end ;
                  left   :  if   x1-length >= 1  then
                            begin
                                 swap;
                                 x1 := x1-length ;
                                 x2 := x2-length ;
                                 swap;
                            end ;
```

```
          end {with case} ;

          with  window_in_used^.window_edge  do
              window ( x1+1,y1+1 , x2-1,y2-1 ) ;

          with  window_in_used^  do
              gotoxy( cursor_pos.x , cursor_pos.y ) ;

     end {if} ;
end;


{ for moving window }


procedure  move_window_to ( x,y : integer ; mode : byte ) ;
var i : integer ;
begin

     if  window_in_used <> main_screen  then
     begin
          if  mode = of_last_window  then
          with  window_in_used^.pred_window^.window_edge  do
          begin
               x := x + x1 ;
               y := y + y1 ;
          end ;

          i := 0 ;
          with  window_in_used^.window_edge  do
          repeat
                inc (i) ;
                if y1 < y then move_window ( down  ,1 ) ;
                if x1 < x then move_window ( right ,1 ) ;
                if y1 > y then move_window ( up    ,1 ) ;
                if x1 > x then move_window ( left  ,1 ) ;

          until  ((y1=y) and (x1=x)) or (i>80)  ;

     end {if with} ;

end {move_window_to} ;


procedure   scan_key_to ( var scan_code , ascii_code  :
char ) ;
var
     scan,ascii   :  char ;
     again        :  boolean ;
begin
```

```
     repeat
           cell.scan_key_to ( scan , ascii ) ;
           again := true ;

           if flag_of_key (ScrollLockOn) then

               case  scan  of
                    UpArrow  :  move_window ( up ,1 ) ;
                   DownArrow :  move_window ( down ,1 ) ;
                   LeftArrow :  move_window ( left ,1 ) ;
                  RightArrow :  move_window ( right ,1 ) ;
                      #$43   :  terminate_to_os_shell ;
               else
                 again := false ;
               end {case}

           else
               again := false
           {end if} ;
     until not again ;
     scan_code  :=  scan  ;
     ascii_code :=  ascii ;
end ;


const  second_key : char = #00 ;

function   ReadKey : char ;
const  null = #00  ;
var    inkey : char ;
       again : boolean ;
begin
     if second_key <> null then
     begin
          ReadKey := second_key ;
          second_key := null ;
          exit ;
     end ;

     repeat
           inkey  :=  Crt.ReadKey ;
           readkey := inkey ;

           if  inkey = null then
           begin
              inkey  :=  Crt.ReadKey ;
              if  (inkey in [#72,#75,#77,#80,#67]) and
                   flag_of_key (ScrollLockOn)  then
              begin
                 again := true ;
                 second_key := null ;
```

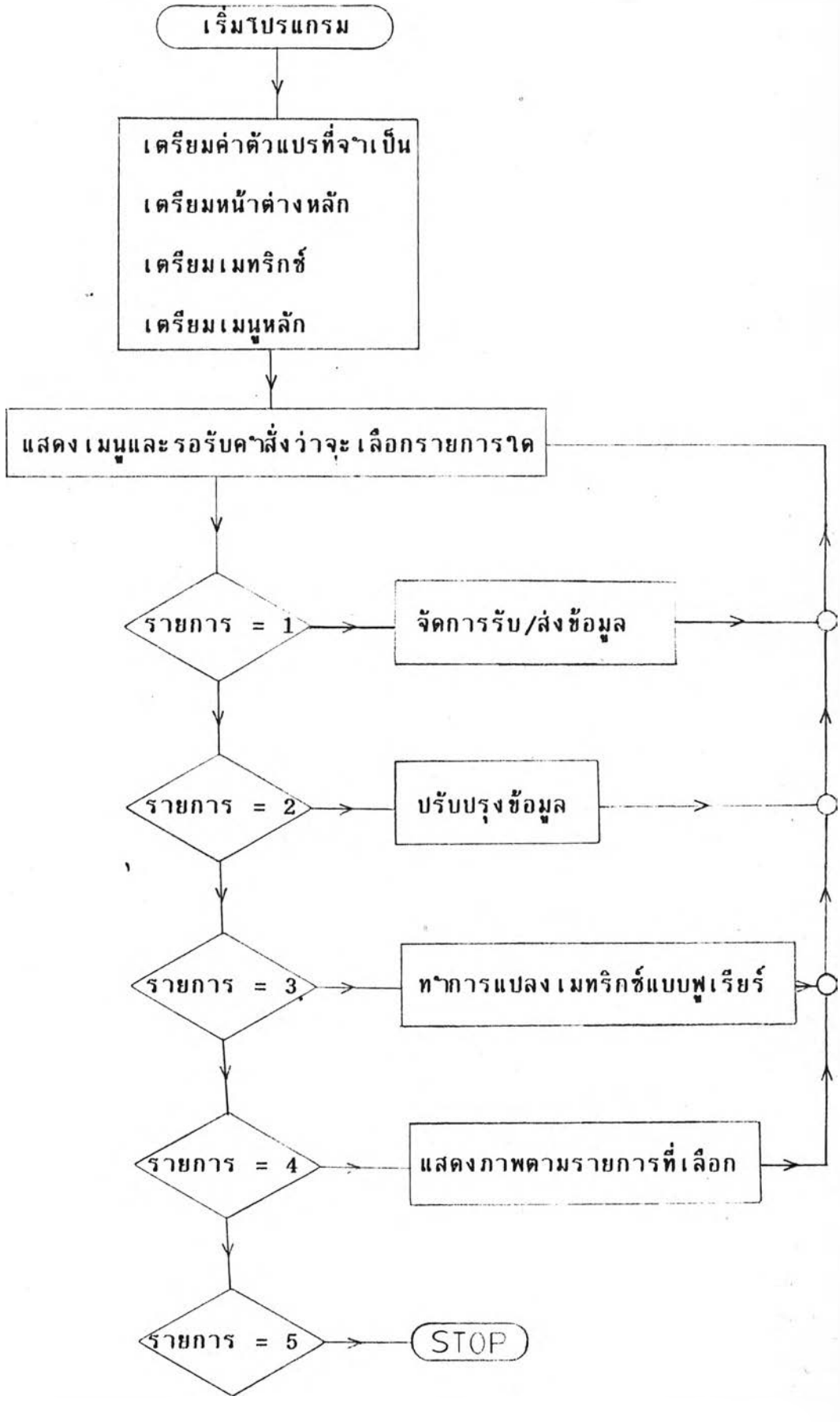
```
                case inkey of
                      #72   :  move_window ( up ,1 ) ;
                      #80   :  move_window ( down ,1 ) ;
                      #75   :  move_window ( left ,1 ) ;
                      #77   :  move_window ( right ,1 )  ;
                      #67   :  terminate_to_os_shell ;
                end {case} ;
             end
             else
             begin
                again := false ;'
                second_key := inkey ;
             end {if}
          end
          else  { inkey <> null }
          begin
             again   := false ;
             second_key := null  ;
          end {if} ;
     until not again ;
end ;


begin
     init_window_system ;
end ..
```

ภาคผนวก ค

โปรแกรมหลัก

ส่วนที่เป็นโปรแกรมหลักนี้จะถูกบันทึกไว้ในแฟ้มข้อมูลชื่อ work.pas ซึ่งจะมีการทำงานดังจะอธิบายได้ตามลำดับขั้น ดังผังงานต่อไปนี้


เริ่มโปรแกรม
เตรียมค่าตัวแปรที่จำเป็น
เตรียมหน้าต่างหลัก
เตรียมเมทริกซ์
เตรียมเมนูหลัก
แสดงเมนูและรอรับคำสั่งว่าจะเลือกรายการใด
รายการ = 1
จัดการรับ/ส่งข้อมูล
รายการ = 2
ปรับปรุงข้อมูล
รายการ = 3
ทำการแปลงเมทริกซ์แบบฟูเรียร์
รายการ = 4
แสดงภาพตามรายการที่เลือก
รายการ = 5
STOP

```
(*                                *)
(*     Filename : WORK.PAS        *)
(*                                *)

program fast_fourier_transforms ;


uses  dos,crt,graph,  { Turbo standard Unit }
      vardef    ,
      music     , {musical instrument}
      timing    , { timeing system}
      cell      , {little tool box}
      glue4     , {connect graphic command to simple command
}
      windsys   , {window manage system}
      menu      , { menu manager}
      getdata   ,
      analizer  ,
      fourier   ,
      display   , {display image}
      option    ;


const
     esc    = #27 ;
     return = #13 ;

     data_file_name = 'c:\mri\sep16-7'  ;


const
     u  :  matrix  = nil ;



procedure  initiate_program ; { declare memory for variable
U : matrix  }
var
     i         :  integer  ;
     memo      :  longint  ;
begin
     VarDef.signal_data_file_name := data_file_name ;

     cr;
     writeln( '    Before create matrix :');
     writeln( '    There are memory available on heap ',
memavail,' bytes . ',
              ' ( ',memavail div 16,' paragraphs ) ') ;
     cr ;

     memo := memavail ;

     new_matrix (u) ;
```

```
     cr;
     writeln( '     After create matrix :');
     writeln( '     There are memory available on heap ',
memavail,' bytes . ',
              ' ( ',memavail div 16,' paragraphs ) ') ;
     cr;cr;
     writeln( '     Use memory for matrix  ',memo-memavail, '
bytes .'
              , '  ( ',(memo-memavail) div 16,' paragraphs
)' ) ;cr;

end {initiate_program}  ;

(*************************************************************
******************)
(*************************************************************
******************)

{  Start main program . }

const
     main_wind    :  window_type = nil ;
     infor_wind   :  window_type = nil ;
     main_menu    :  menu_type   = nil ;

var
     item  : byte    ;
     i,j   : integer ;
     inkey : char    ;

     stop_program : boolean ;

begin
     DirectVideo    := True  ;
     Checkbreak     := False ;

     define_window   ( main_wind , 1,1 , 80,24 ) ;
     window_color    ( main_wind , LightGray , Blue ) ;
     broader_color   ( main_wind , White , Cyan ) ;
     window_heading  ( main_wind , ' NMR lab. ') ;
     head_color      ( main_wind , Yellow , Magenta ) ;

     define_window   ( infor_wind , 1,1 , 21,5 ) ;
     window_color    ( infor_wind , White , Blue ) ;
     broader_color   ( infor_wind , White , Red ) ;
     window_heading  ( infor_wind , ' MNR ') ;
     head_color      ( infor_wind , Yellow , Red ) ;

     open_window     ( main_wind ) ;

     for i := 1 to 25 do cr ;
     highvideo ;
     cr;
```

```
     writeln ('
+============================+ ') ;
     writeln ('                          ¦ Program Fourier
Transform .¦ ') ;
     writeln ('
+============================+ ') ; cr;cr;
     writeln ('    Initiate program . ') ;cr;
     start;
     initiate_program ;
     stop ;
     writeln('    Time used : ',base_60(time_used)) ;
     cr;
     wait;

     close_window ;{and} cancel_window ( main_wind ) ;


     { Redefine window }

     define_window   ( main_wind , 1,1 , 80,10 ) ;
     window_color    ( main_wind , LightGray , Blue ) ;
     broader_color   ( main_wind , White , Cyan ) ;
     window_heading  ( main_wind , ' NMR lab. ') ;
     head_color      ( main_wind , Yellow , Magenta ) ;

     open_window     ( main_wind ) ;


     { create main menu }
     new_menu ( main_menu ) ;
     menu_normal_color    ( main_menu , Yellow , LightGray )
;
     menu_inverse_color   ( main_menu , White , Magenta ) ;
     menu_highlight_color ( main_menu , LightCyan ,LightGray
) ;


     add_menu ( main_menu ,'A','  dAta    ') ;
     add_menu ( main_menu ,'M',' Make up ') ;
     add_menu ( main_menu ,'F','   FFT    ') ;
     add_menu ( main_menu ,'D',' Display ') ;
     add_menu ( main_menu ,'O',' Options ') ;
     add_menu ( main_menu ,'E','  exit    ') ;

     repeat
           clrscr;
           HighVideo;

           gotoxy(24,1); write
('+============================+ ') ;
           gotoxy(24,2); write ('¦ Program Fourier Transform
.¦ ') ;
           gotoxy(24,3); write
('+============================+ ') ;
```

```
            cr;
            write  ('       ');
            case adeptor of
                 CGA    : writeln ( ' Image displayed on color
graphic screen .');
                 EGA    : writeln ( ' Image displayed on
enhance color graphic screen .');
               Hercules : writeln ( ' Image displayed on
monochrome screen .');
                plotter : writeln ( ' Image displayed on HPGL
plotter .');
            end {case} ;


            writeln('         Matrix size : ',max_data,' x
',max_data);


            writeln('         Memory avilable : ',memavail,'
bytes.') ;

            select_menu_bar ( main_menu , item , inkey   , 3,8
) ;

            if  inkey = return  then
              case  item  of
                      1   :  file_data_to ( u )  ;
                      2   :  filtering_matrix ( u ) ;
                      3   :  transform_matrix ( u ) ;
                      4   :  menu_and_display_graph_of ( u ) ;
                    ' 5   :  Select_option ;
              end {case}
            {end if};


         stop_program := (inkey = esc) or (item = 6)  ;

         if  stop_program  then  { confirm to stop }
            begin
                 open_window ( infor_wind ) ; clrscr ;
                 move_window_to ( 59,9 ,relative ) ;
                 cr;
                 writeln ( '   Quit ? (Y/N) ') ;

                 repeat  inkey := readkey ; until inkey in
['y','Y','n','N'] ;

           stop_program := (inkey='y') or (inkey='Y') ;

                 move_window_to ( 1,1 ,of_screen ) ;
                 close_window {infor_wind} ;
            end {if} ;
```

```
      until  stop_program ;

      CloseGraph ;

      textmode ;

      close_window ;{ main_wind }
      cancel_window ( main_wind ) ;
end.

(*****************************************************************
******************)
(*****************************************************************
******************)
(*****************************************************************
******************)
(*****************************************************************
******************)
(*****************************************************************
******************)
.
```

ประวัติผู้เขียน

ร้อยตำรวจตรี วิวัฒน์ สิทธิสรเดช เกิดที่กรุงเทพมหานคร เมื่อวันที่ 8 พฤศจิกายน พ.ศ.2508 สำเร็จการศึกษาเป็นวิทยาศาสตรบัณฑิต สาขาฟิสิกส์ จากมหาวิทยาลัยเชียงใหม่ เมื่อ พ.ศ.2528 เข้าศึกษาต่อในบัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ.2528 เข้ารับราชการตำรวจ รับพระราชทานยศเป็นร้อยตำรวจตรี ได้รับการบรรจุแต่งตั้งให้ดำรงตำแหน่ง รองสารวัตรแผนกตรวจทางเคมีและฟิสิกส์ กองกำกับการ 4 กองพิสูจน์หลักฐาน เมื่อวันที่ 26 มิถุนายน พ.ศ.2529